APPLICATION OF WILCOXON NORM FOR INCREASED OUTLIER INSENSITIVITY IN FUNCTION APPROXIMATION PROBLEMS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & INSTRUMENTATION ENGINEERING

By

NISHANTA SOURAV DAS
Roll No. – 10407029

Under the guidance of

Prof. Ganapati Panda

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela

2008

APPLICATION OF WILCOXON NORM FOR INCREASED OUTLIER INSENSITIVITY IN FUNCTION APPROXIMATION PROBLEMS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & INSTRUMENTATION ENGINEERING

By

NISHANTA SOURAV DAS
Roll No. – 10407029

Under the guidance of

Prof. Ganapati Panda

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela

2008

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the thesis entitled, **"Application of Wilcoxon Norm for increased outlier insensitivity in function approximation problems"** submitted by Sri **Nishanta Sourav Das** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Electronics & Instrumentation Engineering** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Prof. G. Panda
Professor and Head,
Department of Electronics & Communication Engineering,
National Institute of Technology,
Rourkela-769008

Date:

ACKNOWLEDGEMENT

I take this opportunity as a privilege to thank all individuals without whose support and guidance I could not have completed our project in the stipulated period of time. First and foremost I would like to express my deepest gratitude to my Project Supervisor Prof. G. Panda, Head of the Department, Department of Electronics and Communication Engineering, NIT Rourkela for his invaluable support, guidance, motivation and encouragement throughout the period this work was carried out. His readiness for consultation at all times, his educative comments and inputs, his concern and assistance even with practical things have been extremely helpful.

I am grateful to Ms. Babita Majhi , Mr. Jagannath Nanda and Mr. Ajit Kumar Sahoo for their valued suggestions and inputs during the course of the project work.

I would also like to thank all professors and lecturers, and members of the department of Electronics and Communication Engineering for their generous help in various ways for the completion of this thesis. I also extend my thanks to my fellow students for their friendly co-operation.

NISHANTA SOURAV DAS
Roll. No. 10407029
Department of E.C.E.
NIT Rourkela

CONTENTS

ABSTRACT

In system theory, characterization and identification are fundamental problems. When the plant behavior is completely unknown, it may be characterized using certain model and then, its identification may be carried out with some artificial neural networks(ANN) (like multilayer perceptron(MLP) or functional link artificial neural network(FLANN)) or Radial Basis Functions(RBF) using some learning rules such as the back propagation (BP) algorithm. They offer flexibility, adaptability and versatility, for the use of a variety of approaches to meet a specific goal, depending upon the circumstances and the requirements of the design specifications. The first aim of the present thesis is to provide a framework for the systematic design of adaptation laws for nonlinear system identification and channel equalization. While constructing an artificial neural network or a radial basis function neural network, the designer is often faced with the problem of choosing a network of the right size for the task. Using a smaller neural network decreases the cost of computation and increases generalization ability. However, a network which is too small may never solve the problem, while a larger network might be able to. Transmission bandwidth being one of the most precious resources in digital communication, Communication channels are usually modeled as band-limited linear finite impulse response (FIR) filters with low pass frequency response.

The second aim of the thesis is to propose a method of dealing with the inevitable presence of Outliers in system identification and function approximation problems. In statistics, an outlier is an observation that is numerically distant from the rest of the data. Statistics derived from data sets that include outliers may be misleading. As is well known in statistics, the resulting linear regressors by using the rank-based Wilcoxon approach to linear regression problems are usually robust against (or insensitive to) outliers. This is the prime motivation behind the introduction of the Wilcoxon approach to the area of machine learning in this paper. Specifically, we investigate two new learning machines, namely Wilcoxon neural network (WNN) and Wilcoxon generalized radial basis function network (WGRBFN).These provide alternative learning machines when faced with general nonlinear learning problems.

This thesis presents a comprehensive comparative study covering the implementation of Artificial Neural Network (ANN) and Generalized Radial Basis Functions (GRBFNN) and their

Wilcoxon versions, namely Wilcoxon Neural Network (WNN) and Wilcoxon Generalized Radial Basis Function Neural Network (WGRBFNN) for nonlinear system identification and channel equalization. All the structures mentioned above, and their conventional gradient-descent training methods were extensively studied.

Simulation results show that the Wilcoxon learning machines proposed as such have good robustness against outliers as applied to artificial neural networks and generalized radial basis functions.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS USED

ANN	Artificial Neural Network
RBF	Radial Basis Function
GRBFNN	Generalized Radial Basis Function Neural Network
WNN	Wilcoxon Neural Network
WGRBFNN	Wilcoxon Generalized Radial Basis Function Neural Network
BP	Back Propagation
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
ISI	Inter Symbol Interference
LMS	Least Mean Square
MLANN	Multilayer Artificial Neural Network
MLP	Multilayer Perceptron
MSE	Mean Square Error

Chapter 1

INTRODUCTION

1. INTRODUCTION

1.1. INTRODUCTION.

System identification is one of the most important areas in engineering because of its applicability to a wide range of problems. Mathematical system theory, which has in the past few decades evolved into a powerful scientific discipline of wide applicability, deals with analysis and synthesis of systems. The best developed theory for systems defined by linear operators using well established techniques based on linear algebra, complex variable theory and theory of ordinary linear differential equations. Design techniques for dynamical systems are closely related to their stability properties. Necessary and sufficient conditions for stability of linear time-invariant systems have been generated over past century, well-known design methods have been established for such systems. In contrast to this, the stability of nonlinear systems can be established for the most part only on a system-by-system basis.

In the past few decades major advances have been made in adaptive identification and control for identifying and controlling linear time-invariant plants with unknown parameters. The choice of the identifier and the controller structures based on well established results in linear systems theory. Stable adaptive laws for the adjustment of parameters in these which assures the global stability of the relevant overall systems are also based on properties of linear systems as well as stability results that are well known for such systems [1.1].

Machine learning, namely learning from examples, has been an active research area for several decades. Popular and powerful learning machines proposed in the past include artificial neural networks (ANNs) [1]–[4], generalized radial basis function networks (GRBFNs) [5]–[7], fuzzy neural networks (FNNs) [8], [9], and support vector machines (SVMs). They are different in their origins, network configurations, and objective functions. They have also been successfully applied in many branches of science and engineering. In statistical terms, the aforementioned learning machines are nonparametric in the sense that they do not make any assumptions of the functional form, e.g., linearity, of the discriminant or predictive functions. Among these, we would be particularly interested in ANN and GRBFNN.

Robust smoothing is a central idea in statistics that aims to simultaneously estimate and model the underlying structure. Outliers are observations that are separated in some fashion from the rest of the data. Hence, outliers are data points that are not typical of the rest of

the data. Depending on their location, outliers may have moderate to severe effects on the regression model. A regressor or a learning machine is said to be robust if it is insensitive to outliers in the data

1.2. MOTIVATION

Adaptive filtering has proven to be useful in many contexts such as linear prediction, channel equalization, noise cancellation, and system identification. The adaptive filter attempts to iteratively determine an optimal model for the unknown system, or "plant", based on some function of the error between the output of the adaptive filter and the output of the plant. The optimal model or solution is attained when this function of the error is minimized. The adequacy of the resulting model depends on the structure of the adaptive filter, the algorithm used to update the adaptive filter parameters, and the characteristics of the input signal.

When the parameters of a physical system are not available or time dependent it is difficult to obtain the mathematical model of the system. In such situations, the system parameters should be obtained using a system identification procedure. The purpose of system identification is to construct a mathematical model of a physical system from input-output mapping. Studies on linear system identification have been carried out for more than three decades [1.3]. However, identification of nonlinear systems is a promising research area. Nonlinear characteristics such as saturation, dead-zone, etc. are inherent in many real systems. In order to analyze and control such systems, identification of nonlinear system is necessary. Hence, adaptive nonlinear system identification has become more challenging and received much attention in recent years [1.4]. The conventional LMS algorithm [1.5] fails in case of nonlinear channels and plants. Several approaches based on Artificial Neural Network (ANN) and Generalized Radial Basis Functions (GRBFNN) have been discussed in this paper for estimation of nonlinear systems.

In statistics, an outlier is an observation that is numerically distant from the rest of the data. Depending on their location, outliers may have moderate to severe effects on the regression model. Statistics derived from data sets that include outliers may be misleading. For example, if one is calculating the average temperature of 10 objects in a room, and most are between 20 and 25 degrees Celsius, but an oven is at 350 °C, the median of the data may be 23

but the mean temperature will be 55. In this case, the median better reflects the temperature of a randomly sampled object than the mean. Outliers may be indicative of data points that belong to a different population than the rest of the sample set. As is well known in statistics, the resulting linear regressors by using the rank-based Wilcoxon approach to linear regression problems are usually robust against (or insensitive to) outliers. It is then natural to generalize the Wilcoxon approach for linear regression problems to nonparametric Wilcoxon learning machines for nonlinear regression problems. This is the prime motivation behind the introduction of the Wilcoxon approach to the area of machine learning in this paper. Specifically, we investigate two new learning machines, namely Wilcoxon neural network (WNN) and Wilcoxon generalized radial basis function network (WGRBFN).These provide alternative learning machines when faced with general nonlinear learning problems.

1.3. A BRIEF SKETCH OF CONTENTS

- In **Chapter 2**, adaptive modeling and system identification problem is defined for linear and nonlinear plants. The conventional LMS algorithm and other gradient based algorithms for FIR system are derived. Nonlinearity problems are discussed briefly and various methods are proposed for its solution.

- In **Chapter 3**, the theory, structure and algorithms of various artificial neural networks are discussed. We focus on Multilayer Perceptron (MLP).

- **Chapter 4** gives an introduction to Radial Basis Function Neural Network (RBFNN). Different training strategies to train GRBFNN are thoroughly discussed. Simulations are carried out for the stochastic gradient approach for identification of non-linear and noisy plants.

- In **Chapter 5**, introduces the Wilcoxon Learning Approach as applied to various learning machines. The Wilcoxon Norm is defined and its application in developing Wilcoxon Neural Networks (WNN) and Wilcoxon Generalized Radial Basis Functions (WGRBFNN) is shown. The gradient descent methods for these new networks are derived.

- **Chapter 6** summarizes the work done in this thesis work and points to possible directions for future work, more precisely, application of Wilcoxon Norm to various learning machines for increasing the robustness against outliers in various function approximation problems. Simulations are shown with the new update equations as derived and the results are compared with conventional ANN and GRBFNN.

Chapter 2

ADAPTIVE MODELLING AND SYSTEM IDENTIFICATION

2. ADAPTIVE MODELING AND SYSTEM IDENTIFICATION

2.1. INTRODUCTION

Modeling and system identification is a very broad subject, of great importance in the fields of control system, communications, and signal processing. Modeling is also important outside the traditional engineering discipline such as social systems, economic systems, or biological systems. An adaptive filter can be used in modeling that is, imitating the behavior of physical systems which may be regarded as unknown "black boxes" having one or more inputs and one or more outputs. The essential and principal property of an adaptive system is its time-varying, self-adjusting performance. System identification [2.1, 2.2] is the experimental approach to process modeling. System identification includes the following steps :

- ***Experiment design*** Its purpose is to obtain good experimental data and it includes the choice of the measured variables and of the character of the input signals.
- ***Selection of model structure*** A suitable model structure is chosen using prior knowledge and trial and error.
- ***Choice of the criterion to fit***: A suitable cost function is chosen, which reflects how well the model fits the experimental data.
- ***Parameter estimation*** An optimization problem is solved to obtain the numerical values of the model parameters.
- ***Model validation***: The model is tested in order to reveal any inadequacies.

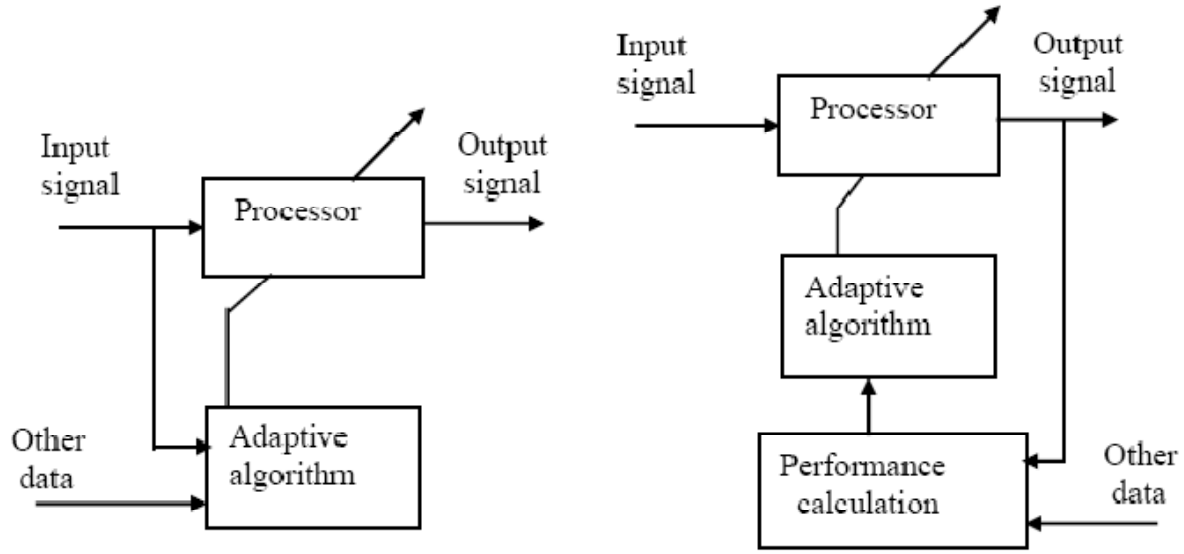
The adaptive systems have following characteristics

1) They can automatically adapt (self-optimize) in the face of changing (non-stationary) environments and changing system requirements.
2) They can be trained to perform specific filtering and decision making tasks.
3) They can extrapolate a model of behavior to deal with new situations after trained on a finite and often small number of training signals and patterns.
4) They can repair themselves to a limited extent.
5) They can be described as nonlinear systems with time varying parameters.

The adaptation is of two types

(i) open-loop adaptation

The open-loop adaptive process is shown in Fig.2.1.(a). It involves making measurements of input or environment characteristics, applying this information to a formula or to a computational algorithm, and using the results to set the adjustments of the adaptive system. The adaptation of process parameters don't depend upon the output signal.



(a) (b)

Fig.2.1. Type of adaptations (a) Open-loop adaptation and (b) Closed-loop adaptation

(ii) closed-loop adaptation

Close-loop adaptation, as shown in Fig. 2.1.(b),on the other hand involves the automatic experimentation with these adjustments and knowledge of their outcome in order to optimize a measured system performance. The latter process may be called adaptation by "performance feedback". The adaptation of process parameters depends upon the input as well as output signal.

2.2. ADAPTIVE FILTER

An adaptive filter [2.3, 2.4] is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. Adaptive filters are often realized either as a set of program instructions running on an arithmetical processing device such as a microprocessor or digital signal processing (DSP) chip, or as a set of logic operations implemented in a field-programmable gate array (FPGA). However, ignoring any errors

introduced by numerical precision effects in these implementations, the fundamental operation of an adaptive filter can be characterized independently of the specific physical realization that it takes. For this reason, we shall focus on the mathematical forms of adaptive filters as opposed to their specific realizations in software or hardware. An adaptive filter is defined by four aspects:
1. The signals being processed by the filter.

2. The structure that defines how the output signal of the filter is computed from its input signal

3. The parameters within this structure that can be iteratively changed to alter the filter's input-output relationship.

4. The adaptive algorithm that describes how the parameters are adjusted from one time instant to the next.

By choosing a particular adaptive filter structure, one specifies the number and type of parameters that can be adjusted. The adaptive algorithm used to update the parameter values of the system can take on an infinite number of forms and is often derived as a form of optimization procedure that minimizes an error.

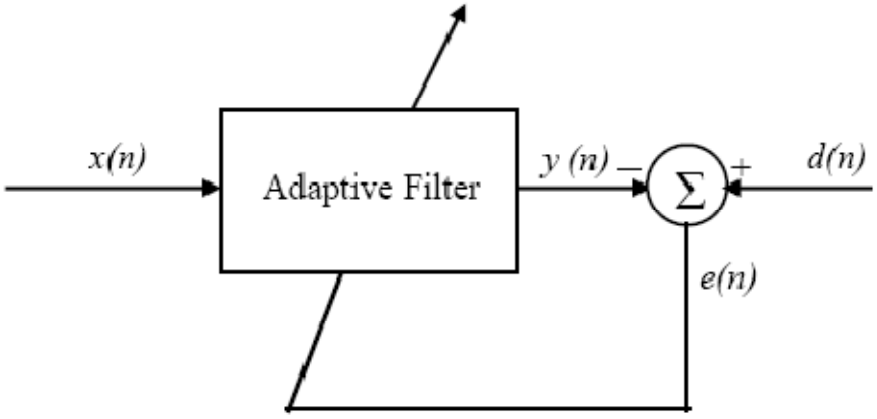


Fig.2.2. General Adaptive Filtering

Fig.2.2. shows a block diagram in which a sample from a digital input signal *x(n)* is fed into a device, called an adaptive filter, that computes a corresponding output signal sample *y(n)* at time

n. For the moment, the structure of the adaptive filter is not important, except for the fact that it contains adjustable parameters whose values affect how *y(n)* is computed. The output signal is compared to a second signal *d(n),* called the desired response signal, by subtracting the two samples at time *n.* This difference signal, given by

$$e(n) = d(n) - y(n) \tag{2.1}$$

is known as the error signal. The error signal is fed into a procedure which alters or adapts the parameters of the filter from time *n* to time *(n* + 1) in a well-defined manner. As the time index *n* is incremented, it is hoped that the output of the adaptive filter becomes a better and better match to the desired response signal through this adaptation process, such that the magnitude of *e(n)* decreases over time. In the adaptive filtering task, adaptation refers to the method by which the parameters of the system are changed from time index *n* to time index *(n* +1). The number and types of parameters within this system depend on the computational structure chosen for the system. We now discuss different filter structures that have been proven useful for adaptive filtering tasks.

2.3. FILTER STRUCTURES

In general, any system with a finite number of parameters that affect how *y(n)* is computed from *x(n)* could be used for the adaptive filter in Fig. 2.2.. Define the parameter or coefficient vector

$$W(n) = [\, w_0(n) \;\; w_1(n) \;\ldots\ldots\; w_{L-1}(n) \,]^T \tag{2.2}$$

where $\{ w_i(n)\}$, $0 < i < L - 1$ are the L parameters of the system at time n.

The filter model typically takes the form of a finite-impulse-response (FIR) or infinite-impulse-response (IIR) filter. Figure2.3. shows the structure of a direct-form FIR filter, also known as a tapped-delay-line or transversal filter, where z^{-1} denotes the unit delay element and each $w_i(n)$ is a multiplicative gain within the system. In this case, the parameters in $W(n)$ correspond to the impulse response values of the filter at time n. We can write the output signal $y(n)$ as

$$\begin{aligned} y(n) &= \sum_{i=0}^{L-1} w_i(n)x(n-i) \\ &= W^T(n)X(n) \end{aligned} \tag{2.3}$$

where $X(n) = [\, x(n) \;\; x(n-1) \;\ldots\ldots\; x(n-L+1) \,]^T$ denotes the input signal vector and ^{T} denotes vector transpose. Note that this system requires L multipliers and L - 1 delays to implement and

these computations are easily performed by a processor or circuit so long as *L* is not too large and the sampling period for the signals is not too short. It also requires a total of *2L* memory locations to store the *L* input signal samples and the *L* coefficient values, respectively.

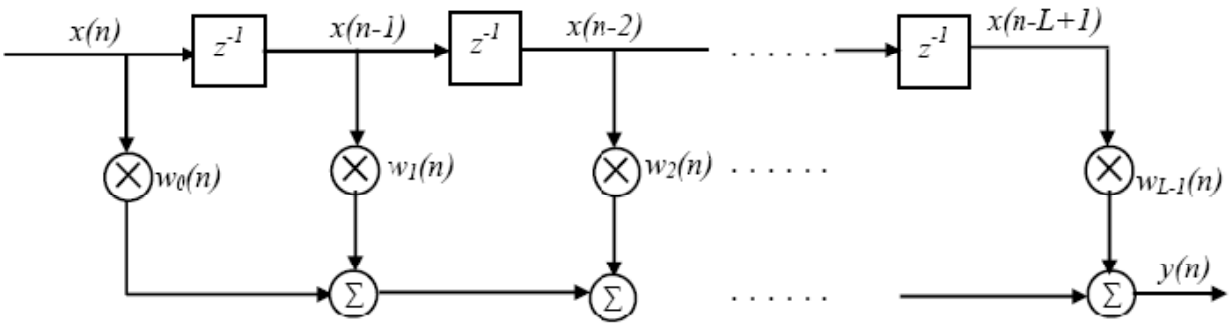


Fig.2.3. FIR filter structure

2.4. APPLICATION OF ADAPTIVE FILTERS.

Perhaps the most important driving forces behind the developments in adaptive filters throughout their history have been the wide range of applications in which such systems can be used. We now discuss the forms of these applications in terms of more-general problem classes that describe the assumed relationship between *d(n)* and *x(n).* Our discussion illustrates the key issues in selecting an adaptive filter for a particular task.

2.4.1. Direct Modeling (Function Approximation & System Identification)

In function approximation problems, we are given a set of input-output patterns and we try to estimate the underlying function that relates the input to the output. This is done by passing the same set of input points to the function and an adaptive filter kept parallel to the function,Fig.2.4 gives an illustration. The outputs or response of both the function and the filter is found out and their difference is noted. This difference is the error. The error is minimized by an adaptive algorithm that updates the weights of the adaptive filter. System Identification is a special case of function approximation. Here the underlying function is the provided by a plant or system and our aim is to determine the impulse response of this system.

In direct modeling, the adaptive model is kept parallel with the unknown plant. Modeling a single-input, single-output system is illustrated in Fig.2.5..Both the unknown system and adaptive filter are driven by the same input. The adaptive filter adjusts itself in such a way that its output is

matched with that of the unknown system. Upon convergence, the structure and parameter values of the adaptive system may or may not resemble those of unknown systems, but the input-output response relationship will match. In this sense, the adaptive system becomes a model of the unknown plant

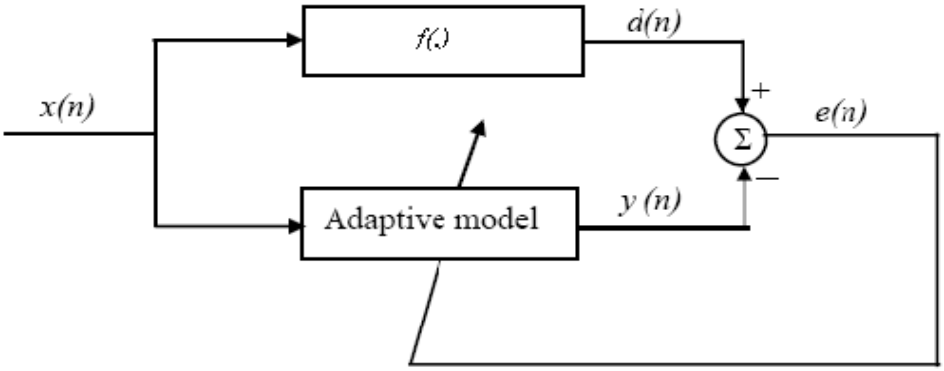


Fig.2.4. Function approximation

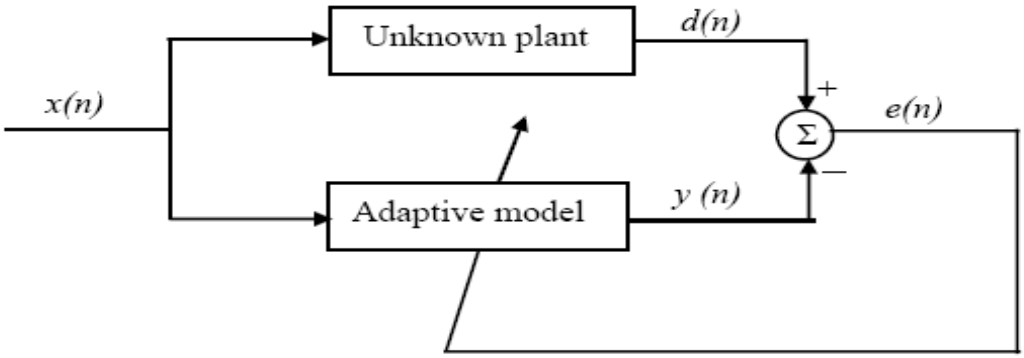


Fig.2.5. System Identification

Let *d(n)* and *y(n)* represent the output of the unknown system and adaptive model with *x(n)* as its input. Here, the task of the adaptive filter is to accurately represent the signal *d(n)* at its output. If *y(n) = d (n)*, then the adaptive filter has accurately modeled or identified the portion of the unknown system that is driven by *x(n)*.

Since the model typically chosen for the adaptive filter is a linear filter, the practical goal of the adaptive filter is to determine the best linear model that describes the input-output relationship of the unknown system. Such a procedure makes the most sense when the unknown system is also a linear model of the same structure as the adaptive filter, as it is possible that *y(n) = d(n)* for some set of adaptive filter parameters. For ease of discussion, let the unknown system and the adaptive filter both be FIR filters, such that

$d(n) = W^T_{OPT}(n)X(n)$ (2.4)

where $W_{OPT}(n)$ is an optimum set of filter coefficients for the unknown system at time n. In this problem formulation, the ideal adaptation procedure would adjust $W(n)$ such that $W(n) = W_{OPT}(n)$ as $n \rightarrow \infty$. In practice, the adaptive filter can only adjust $W(n)$ such that $y(n)$ closely approximates $d(n)$ over time.

The system identification task is at the heart of numerous adaptive filtering applications. We list several of these applications here

- Plant Identification
- Echo Cancellation for Long-Distance Transmission
- Acoustic Echo Cancellation
- Adaptive Noise Canceling

2.4.2. Inverse Modeling

We now consider the general problem of inverse modeling, as shown in Fig.2.6. In this diagram, a source signals $s(n)$ is fed into a plant that produces the input signal $x(n)$ for the adaptive filter. The output of the adaptive filter is subtracted from a desired response signal that is a delayed version of the source signal, such that

$d(n) = s(n - \Delta)$ (2.5)

where Δ is a positive integer value. The goal of the adaptive filter is to adjust its characteristics such that the output signal is an accurate representation of the delayed source signal.

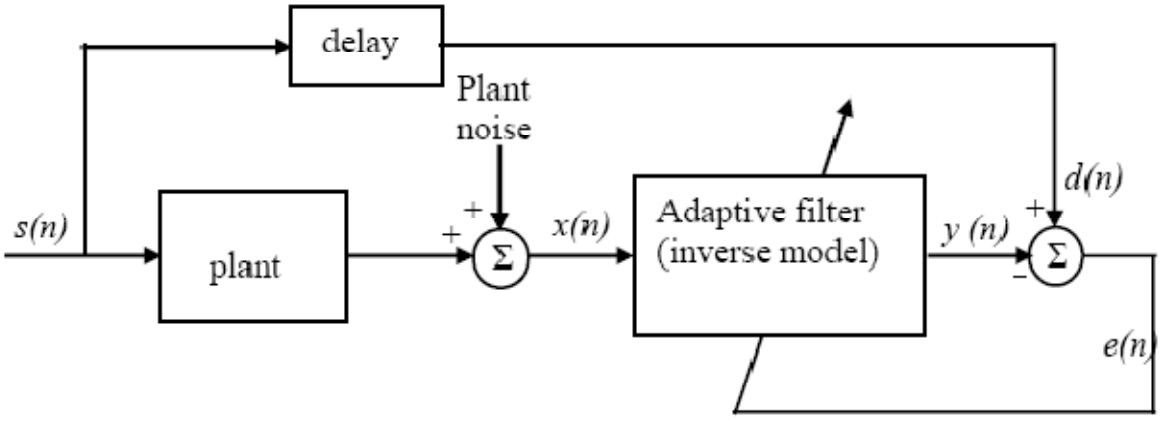


Fig.2.6 Inverse Modeling

2.5. GRADIENT BASED ADAPTIVE ALGORITHM

An adaptive algorithm is a procedure for adjusting the parameters of an adaptive filter to minimize a cost function chosen for the task at hand. In this section, we describe the general form of many adaptive FIR filtering algorithms and present a simple derivation of the LMS adaptive algorithm. In our discussion, we only consider an adaptive FIR filter structure, such that the output signal *y(n)* is given by (2.3). Such systems are currently more popular than adaptive IIR filters because

(1) The input-output stability of the FIR filter structure is guaranteed for any set of fixed coefficients, and

(2) The algorithms for adjusting the coefficients of FIR filters are simpler in general than those for adjusting the coefficients of IIR filters.

2.5.1. General Form of Adaptive FIR Algorithm

The general form of an adaptive FIR filtering algorithm is

$$W(n+1) = W(n) + \mu(n)\, G\,(\, e(n), X(n), \varphi(n)\,) \qquad (2.6)$$

where G(-) is a particular vector-valued nonlinear function, $\mu(n)$ is a step size parameter, $e(n)$ and $X(n)$ are the error signal and input signal vector, respectively, and $\varphi(n)$ is a vector of states that store pertinent information about the characteristics of the input and error signals and/or the coefficients at previous time instants. In the simplest algorithms, $\varphi(n)$ is not used, and the only information needed to adjust the coefficients at time *n* are the error signal, input signal vector, and step size.

The step size is so called because it determines the magnitude of the change or "step" that is taken by the algorithm in iteratively determining a useful coefficient vector. Much research effort has been spent characterizing the role that $\mu(n)$ plays in the performance of adaptive filters in terms of the statistical or frequency characteristics of the input and desired response signals. Often, success or failure of an adaptive filtering application depends on how the value of $\mu(n)$ is chosen or calculated to obtain the best performance from the adaptive filter.

2.5.2. The Mean-Squared Error Cost Function

The form of G(-) in (2.6) depends on the cost function chosen for the given adaptive filtering task. We now consider one particular cost function that yields a popular adaptive algorithm. Define the mean-squared error (MSE) cost function as

$$\xi_{MSE}(n) = \frac{1}{2}\int_{-\infty}^{\infty} e^2(n) p_n\big(e(n)\big)\, de(n)$$

$$= \frac{1}{2} E\{e^2(n)\} \qquad (2.7)$$

where $p_n(e(n))$ represents the probability density function of the error at time n and $E\{\cdot\}$ is shorthand for the *expectation integral* on the right-hand side of (2.7). The MSE cost function is useful for adaptive FIR filters because

• $\xi_{MSE}(n)$ has a well-defined minimum with respect to the parameters in $W(n)$;

• the coefficient values obtained at this minimum are the ones that minimize the power in the error signal $e(n)$, indicating that $y(n)$ has approached $d\{n)$; and

• ξ_{MSE} is a smooth function of each of the parameters in $W(n)$, such that it is differentiable with respect to each of the parameters in $W(n)$. The third point is important in that it enables us to determine both the optimum coefficient values given knowledge of the statistics *of d(n)* and *x(n)* as well as a simple iterative procedure for adjusting the parameters of an FIR filter.

2.5.3. The Wiener Solution.

For the FIR filter structure, the coefficient values in $W(n)$ that minimize $\xi_{MSE}(n)$ are well-defined if the statistics of the input and desired response signals are known. The formulation of this problem for continuous-time signals and the resulting solution was first derived by Wiener [2.3]. Hence, this optimum coefficient vector $W_{MSE}(n)$ is often called the *Wiener solution* to the adaptive filtering problem. The extension of Wiener's analysis to the discrete-time case is attributed to Levinson . To determine $W_{MSE}(n)$ we note that the function $\xi_{MSE}(n)$ in (2.7) is quadratic in the parameters $\{w_i(n)\}$, and the function is also differentiable. Thus, we can use a result from optimization theory that states that the derivatives of a smooth cost function with respect to each of the parameters is zero at a minimizing point on the cost function error surface. Thus, $W_{MSE}(n)$ can be found from the solution to the system of equations

$$\frac{\partial \xi_{MSE}(n)}{\partial w_i(n)} = 0 \quad , 0 \leq i \leq L\text{-}1 \qquad (2.8)$$

Taking derivatives of $\xi_{MSE}(n)$ in (2.7) we obtain

$$\frac{\partial \xi_{MSE}(n)}{\partial w_i(n)} = E\{ e(n) \frac{\partial e(n)}{\partial w_i(n)} \} \quad (2.9)$$

$$= -E\{ e(n) \frac{\partial y(n)}{\partial w_i(n)} \} \quad (2.10)$$

$$= -E\{ e(n)\, x(n-i) \} \quad (2.11)$$

$$= -(E\{ d(n)\, x(n-i) \} - \sum_{j=0}^{L-1} E\{ x(n-i)x(n-j)\} w_i(n) \quad (2.12)$$

where we have used the definitions of *e(n)* and of *y(n)* for the FIR filter structure in (2.1) and (2.6), respectively, to expand the last result in (2.15). By defining the matrix $R_{XX}(n)$ (autocorrelation matrix) and vector $P_{dx}(n)$ (cross correlation matrix) as

$$R_{XX}(n) = E (X(n)X(n)^T)$$

$$and \quad (2.13)$$

$$P_{dx}(n) = E (d(n) X(n))$$

respectively, we can combine (2.8) and (2.13) to obtain the system of equations in vector form as

$$R_{XX}(n)\, W_{MSE}(n) - P_{dx}(n) = 0 \quad (2.14)$$

where 0 is the zero vector. Thus, so long as the matrix $R_{XX}(n)$ is invertible, the optimum Wiener solution vector for this problem is

$$W_{MSE}(n) = R_{XX}^{-1}(n)\, P_{dx}(n) \quad (2.15)$$

2.5.4. The Method of Steepest Descent

The method of steepest descent is a celebrated optimization procedure for minimizing the value of a cost function $\xi(n)$ with respect to a set of adjustable parameters $W(n)$. This procedure adjusts each parameter of the system according to

$$w_i(n+1) = w_i(n) + \mu(n)\frac{\partial \xi(n)}{\partial w_i(n)} \tag{2.16}$$

In other words, the i^{th} parameter of the system is altered according to the derivative of the cost function with respect to the i^{th} parameter. Collecting these equations in vector form, we have

$$W(n+1) = W(n) + \mu(n)\frac{\partial \xi(n)}{\partial W(n)} \tag{2.17}$$

where $\partial\xi(n)/\partial W(n)$ is a vector of derivatives $d\xi(n)/dw_i(n)$.

Substituting these results into (2.17) yields the update equation for $W(n)$ as

$$W(n+1) = W(n) + \mu(n)(P_{dx}(n) - R_{XX}(n)W(n)) \tag{2.18}$$

However, this steepest descent procedure depends on the statistical quantities $E\{d(n)x(n\text{-i})\}$ and $E\{x(n\text{-}i)x(n\text{-}j)\}$ contained in $P_{dx}(n)$ and $R_{xx}(n)$, respectively. In practice, we only have measurements of both $d(n)$ and $x(n)$ to be used within the adaptation procedure. While suitable estimates of the statistical quantities needed for (2.21) could be determined from the signals $x(n)$ and $d\{n)$, we instead develop an approximate version of the method of steepest descent that depends on the signal values themselves. This procedure is known as the LMS (least mean square) algorithm.

2.6. LMS ALGORITHM

The cost function $\xi(n)$ chosen for the steepest descent algorithm of (2.16) determines the coefficient solution obtained by the adaptive filter. If the MSE cost function in (2.7) is chosen, the resulting algorithm depends on the statistics of $x(n)$ and $d(n)$ because of the expectation operation that defines this cost function. Since we typically only have measurements of $d(n)$ and

of $x(n)$ available to us, we substitute an alternative cost function that depends only on these measurements.

we can propose the simplified cost function $\xi_{LMS}(n)$given by

$$\xi_{LMS}(n) = \frac{1}{2} e^2(n) \qquad (2.19)$$

This cost function can be thought of as an instantaneous estimate of the MSE cost function, as $\xi_{MSE}(n) = E\{ \xi_{LMS}(n) \}$. Although it might not appear to be useful, the resulting algorithm obtained when $\xi_{LMS}(n)$ is used for $\xi(n)$ in (2.16) is extremely useful for practical applications. Taking derivatives of $\xi_{LMS}(n)$ with respect to the elements of $W(n)$ and substituting the result into (2.16), we obtain the LMS adaptive algorithm given by

$$W(n+1) = W(n) + \mu(n)e(n)\, X(n) \qquad (2.20)$$

Equation (2.20) requires only multiplications and additions to implement. In fact, the number and type of operations needed for the LMS algorithm is nearly the same as that of the FIR filter structure with fixed coefficient values, which is one of the reasons for the algorithm's popularity. The behavior of the LMS algorithm has been widely studied, and numerous results concerning its adaptation characteristics under different situations have been developed. For now, we indicate its useful behavior by noting that the solution obtained by the LMS algorithm near its convergent point is related to the Wiener solution. In fact, analysis of the LMS algorithm under certain statistical assumptions about the input and desired response signals show that

$$\lim_{n\to\infty} E\{ W(n)\} = W_{MSE}\,(n) \qquad (2.21)$$

when the Wiener solution $W_{MSE}(n)$ is a fixed vector. Moreover, the average behavior of the LMS algorithm is quite similar to that of the steepest descent algorithm in (2.18) that depends explicitly on the statistics of the input and desired response signals. In effect, the iterative nature of the LMS coefficient updates is a form of time-averaging that smoothes the errors in the instantaneous gradient calculations to obtain a more reasonable estimate of the true gradient.The problem is that gradient descent is a local optimization technique, which is limited because it is

unable to converge to the global optimum on a multimodal error surface if the algorithm is not initialized in the basin of attraction of the global optimum.

Several modifications exist for gradient based algorithms in attempt to enable them to overcome local optima. One approach is to simply add a momentum term [2.3] to the gradient computation of the gradient descent algorithm to enable it to be more likely to escape from a local minimum. This approach is only likely to be successful when the error surface is relatively smooth with minor local minima, or some information can be inferred about the topology of the surface such that the additional gradient parameters can be assigned accordingly. Other approaches attempt to transform the error surface to eliminate or diminish the presence of local minima [2.16], which would ideally result in a unimodal error surface. The problem with these approaches is that the resulting minimum transformed error used to update the adaptive filter can be biased from the true minimum output error and the algorithm may not be able to converge to the desired minimum error condition. These algorithms also tend to be complex, slow to converge, and may not be guaranteed to emerge from a local minimum.

Another approach, attempts to locate the global optimum by running several LMS algorithms in parallel, initialized with different initial coefficients. The notion is that a larger, concurrent sampling of the error surface will increase the likelihood that one process will be initialized in the global optimum valley. This technique does have potential, but it is inefficient and may still suffer the fate of a standard gradient technique in that it will be unable to locate the global optimum. By using a similar congregational scheme, but one in which information is collectively exchanged between estimates and intelligent randomization is introduced, structured stochastic algorithms are able to hill-climb out of local minima. This enables the algorithms to achieve better, more consistent results using a fewer number of total estimate.

2.7. SYSTEM IDENTIFICATION

System identification concerns with the determination of a system, on the basis of input output data samples. The identification task is to determine a suitable estimate of finite dimensional parameters which completely characterize the plant. The selection of the estimate is based on comparison between the actual output sample and a predicted value on the basis of input data up to that instant. An adaptive automaton is a system whose structure is alterable or adjustable in such a way that its behavior or performance improves through contact with its

environment. Depending upon input-output relation, the identification of systems can have two groups

A. Static System Identification

In this type of identification the output at any instant depends upon the input at that instant. These systems are described by the algebraic equations. The system is essentially a memoryless one and mathematically it is represented as $y(n) = f[x(n)]$ where $y(n)$ is the output at the nth instant corresponding to the input $x(n)$.

B. Dynamic System Identification

In this type of identification the output at any instant depends upon the input at that instant as well as the past inputs and outputs. Dynamic systems are described by the difference or differential equations. These systems have memory to store past values and mathematically represented as $y(n)=f[x(n), x(n-1),x(n-2)\ldots\ldots\ldots y(n-1),y(n-2),\ldots\ldots]$ where $y(n)$ is the output at the nth instant corresponding to the input $x(n)$.

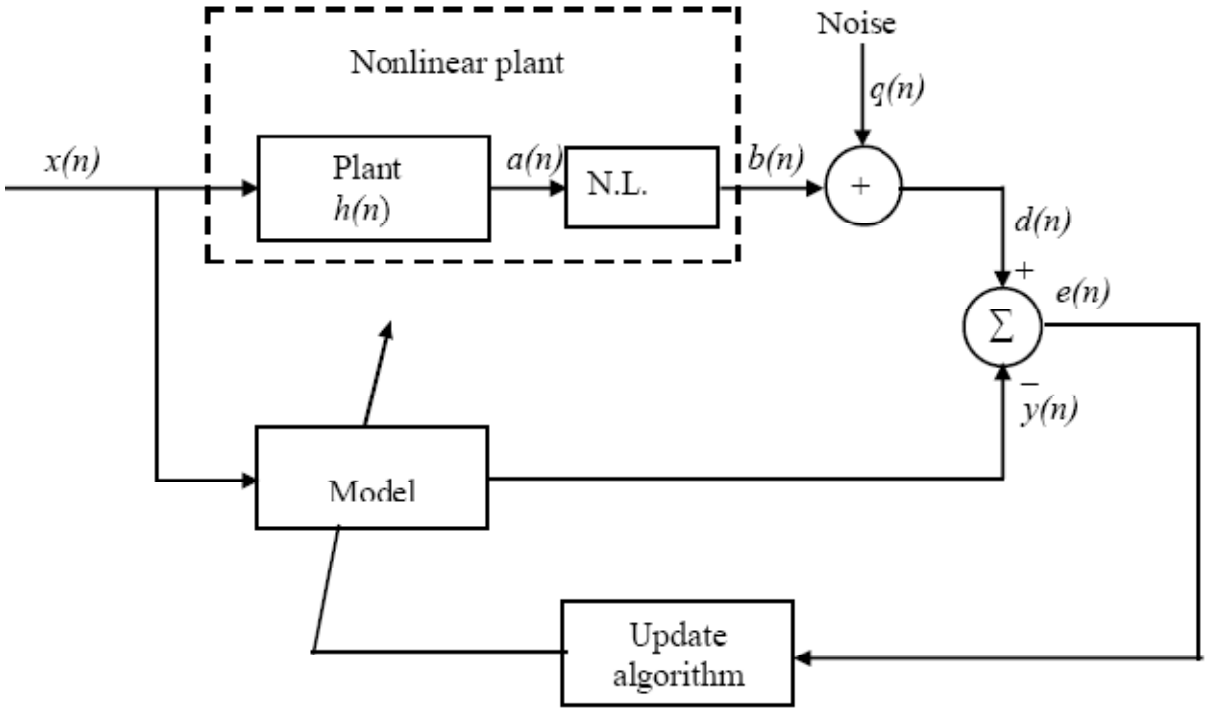


Fig.2.7. Block Diagram of System Identification

A system identification structure is shown in Fig.2.6. The model is placed parallel to the nonlinear plant and same input is given to the plant as well as the model. The impulse response of the linear segment of the plant is represented by *h(n)* which is followed by nonlinearity(NL) associated with it. White Gaussian noise *q(n)* is added with nonlinear output accounts for measurement noise. The desired output *d(n)* is compared with the estimated output *y(n*) of the identifier to generate the error *e(n)* which is used by some adaptive algorithm for updating the weights of the model. The training of the filter weights is continued until the error becomes minimum and does not decrease further. At this stage the correlation between input signal and error signal is minimum. Then the training is stopped and the weights are stored for testing. For testing purpose new samples are passed through both the plant and the model and their responses are compared.

Chapter 3

ARTIFICIAL NEURAL NETWORKS

2. ARTIFICIAL NEURAL NETWORK

3.1. INTRODUCTION

Because of nonlinear signal processing and learning capability, Artificial Neural Networks (ANN's) have become a powerful tool for many complex applications including functional approximation, nonlinear system identification and control, pattern recognition and classification, and optimization. The ANN's are capable of generating complex mapping between the input and the output space and thus, arbitrarily complex nonlinear decision boundaries can be formed by these networks. An artificial neuron basically consists of a computing element that performs the weighted sum of the input signal and the connecting weight. The sum is added with the bias or threshold and the resultant signal is then passed through a non-linear element of *tanh(.)* type. Each neuron is associated with three parameters whose learning can be adjusted; these are the connecting weights, the bias and the slope of the non-linear function. For the structural point of view a neural network(NN) may be single layer or it may be multi-layer. In multi-layer structure, there is one or many artificial neurons in each layer and for a practical case there may be a number of layers. Each neuron of the one layer is connected to each and every neuron of the next layer.

A neural network is a massively parallel distributed processor made up of simple processing unit, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two types

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Artificial Neural Networks (ANN) has emerged as a powerful learning technique to perform complex tasks in highly nonlinear dynamic environments. Some of the prime advantages of using ANN models are their ability to learn based on optimization of an appropriate error function and their excellent performance for approximation of nonlinear function. At present, most of the work on system identification using neural networks are based on multilayer feed forward neural networks with back propagation learning or more efficient variations of this algorithm On the other hand the Functional link ANN(FLANN) originally proposed by Paois a single layer structure with functionally mapped inputs. The performance of FLANN for system

identification of nonlinear systems has been reported [3.5] in the literature. Patra and Kot have used Chebyschev expansions for nonlinear system identification and have shown that the identification performance is better than that offered by the multilayer ANN (MLANN) model. Wang and Chen have presented a fully automated recurrent neural network (FARNN) that is capable of self-structuring its network in a minimal representation with satisfactory performance for unknown dynamic system identification and control

3.2. SINGLE NEURON STRUCTURE

In 1958, Rosenblatt demonstrated some practical applications using the perceptron [3.8]. The perceptron is a single level connection of McCulloch-Pitts neurons sometimes called single-layer feed forward network*s*. The network is capable of linearly separating the input vectors into pattern of classes by a hyper plane. A linear associative memory is an example of a single-layer neural network. In such an application, the network associates an output pattern (vector) with an input pattern (vector), and information is stored in the network by virtue of modifications made to the synaptic weights of the network.

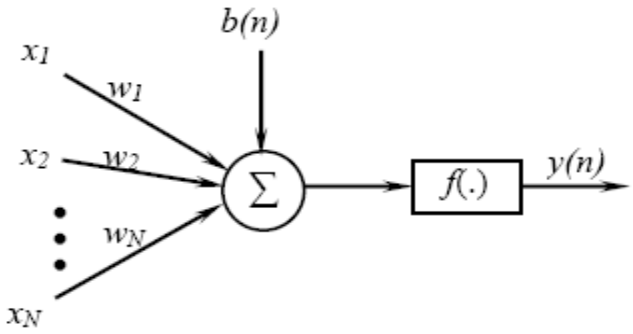


Fig. 3.1. A single Neuron

The structure of a single neuron is presented in Fig. 3.1.An artificial neuron involves the computation of the weighted sum of inputs and threshold [3.9, 3.10]. The resultant signal is then passed through a non-linear activation function. The output of the neuron may be represented as,

$$y(n) = f\left[\sum_{i=1}^{N} w_j(n)\, x_j(n) + b(n)\right] \tag{3.1}$$

Where $b(n)$ = threshold to the neuron is called as bias.

$w_j(n)$ = weight associated with the j^{th} input, and N = no. of inputs to the neuron.

3.2.1. Activation Functions and Bias.

The perceptron internal sum of the inputs is passed through an activation function, which can be any monotonic function. Linear functions can be used but these will not contribute to a non-linear transformation within a layered structure, which defeats the purpose of using a neural filter implementation. A function that limits the amplitude range and limits the output strength of each perceptron of a layered network to a defined range in a non-linear manner will contribute to a nonlinear transformation. There are many forms of activation functions, which are selected according to the specific problem. All the neural network architectures employ the activation function [3.1, 3.8] which defines as the output of a neuron in terms of the activity level at its input (ranges from -1 to 1 or 0 to 1). Table 3.1 summarizes the basic types of activation functions. The most practical activation functions are the sigmoid and the hyperbolic tangent functions. This is because they are differentiable.

The bias gives the network an extra variable and the networks with bias are more powerful than those of without bias. The neuron without a bias always gives a net input of zero to the activation function when the network inputs are zero. This may not be desirable and can be avoided by the use of a bias.

Table 3.1 COMMON ACTIVATION FUNCTIONS

Name	**Definition**
Linear	$f(x) = kx$
Step	$f(x) = \beta, \quad if\ x \geq k$ $= \delta, \quad if\ x < k$
Sigmoid	$f(x) = \frac{1}{1+e^{-\alpha x}}, \alpha > 0$
Hyperbolic Tangent	$f(x) = \tanh(\gamma x) = \frac{1-e^{-\gamma x}}{1+e^{-\gamma x}}, \gamma > 0$
Gaussian	$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$

3.2.2 Learning Processes

The property that is of primary significance for a neural network is that the ability of the network to learn from its environment, and to improve its performance through learning. The improvement in performance takes place over time in accordance with some prescribed measure. A neural network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels. Ideally, the network becomes more knowledgeable about its environment after each iteration of learning process. Hence we define learning as:

"It is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded."

The processes used are classified into two categories as described in [3.1]:

(A) Supervised Learning (Learning With a Teacher)

(B) Unsupervised Learning (Learning Without a Teacher)

(A) Supervised Learning:

We may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment is, however unknown to neural network of interest. Suppose now the teacher and the neural network are both exposed to a training vector, by virtue of built-in knowledge, the teacher is able to provide the neural network with a desired response for that training vector. Hence the desired response represents the optimum action to be performed by the neural network. The network parameters such as the weights and the thresholds are chosen arbitrarily and are updated during the training procedure to minimize the difference between the desired and the estimated signal. This updation is carried out iteratively in a step-by-step procedure with the aim of eventually making the neural network emulate the teacher. In this way knowledge of the environment available to the teacher is transferred to the neural network. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself. This is the form of supervised learning.

The update equations for weights are derived as LMS:

$$w_j(n+1) = w_j(n) + \mu(n)\Delta w_j(n) \tag{3.2}$$

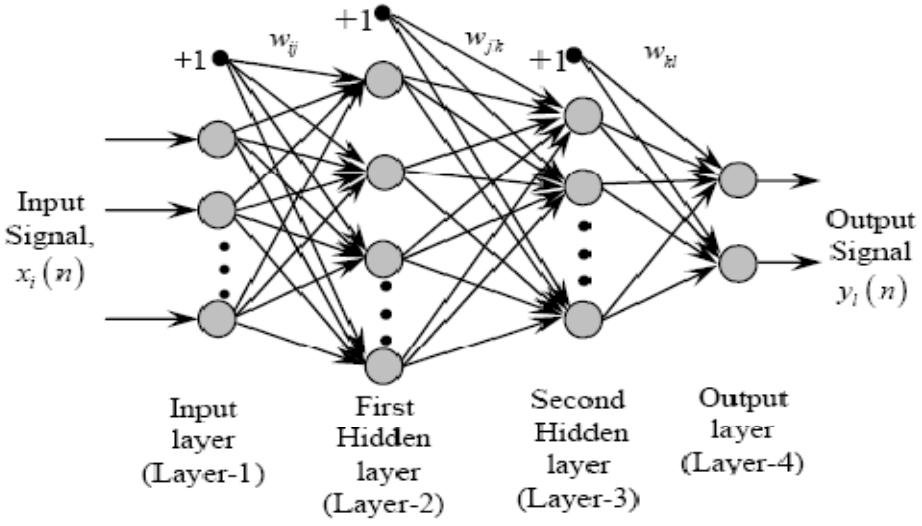
$\Delta w_j(n)$ is the change in w_j in nth iteration.

(B) Unsupervised Learning:

In unsupervised learning or self-supervised learning there is no teacher to over-see the learning process, rather provision is made for a task independent measure of the quantity of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become turned to the statistical regularities of the input data, it develops the ability to form the internal representations for encoding features of the input and thereby to create new classes automatically. In this learning the weights and biases are updated in response to network input only. There are no desired outputs available. Most of these algorithms perform some kind of clustering operation. They learn to categorize the input patterns into some classes.

3.3. MULTILAYER PERCEPTRON

In the multilayer perceptron (MLP), the input signal propagates through the network in a forward direction, on a layer-by-layer basis. This network has been applied successfully to solve some difficult problems by training in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm [3.1,3.9]. The scheme of MLP using four layers is shown in Fig.3.2. $x_i(n)$ represent the input to the network, f_i and f_k represent the output of the two hidden layers and $y_l(n)$ represents the output of the final layer of the neural network. The connecting weights between the input to the first hidden layer, first to second hidden layer and the second hidden layer to the output layers are represented by w_{ij}, w_{jk}, w_{kl} respectively.



3.2 MLP network

If P_1 is the number of neurons in the first hidden layer, each element of the output vector of first hidden layer may be calculated as,

$$f_i = \varphi_f \left[\sum_{i=1}^{N} w_{ij} x_i(n) + b_j \right] \; i = 1,2,3, \ldots N, j = 1,2,3, .. P_1 \tag{3.3}$$

where b_j is the threshold to the neurons of the first hidden layer, N is the no. of inputs and $\phi(.)$ is the nonlinear activation function in the first hidden layer chosen from the Table 3.1. The time index n has been dropped to make the equations simpler. Let P_2 be the number of neurons in the second hidden layer. The output of this layer is represented as, f_k and may be written as

$$f_k = \varphi_k \left[\sum_{k=1}^{p_1} w_{jk} f_j + b_k \right], k = 1,2,3, \ldots P_2 \tag{3.4}$$

where, b_k is the threshold to the neurons of the second hidden layer. The output of the final output layer can be calculated as

$$y_l(n) = \varphi_l \left[\sum_{k=1}^{p_2} w_{kl} f_k + b_l \right], l = 1,2,3, \ldots P_3 \tag{3.5}$$

where, α_l is the threshold to the neuron of the final layer and P_3 is the no. of neurons in the output layer. The output of the MLP may be expressed as

$$y_l(n) = \varphi_n \left[\sum_{k=1}^{p_2} w_{kl} \varphi_k \left(\sum_{j=1}^{p_1} w_{jk} \, \varphi_j \left\{ \sum_{i=1}^{N} w_{ij} x_i(n) + b_j \right\} + b_k \right) + b_l \right] \tag{3.6}$$

3.3.1. Backpropagation Algorithm.

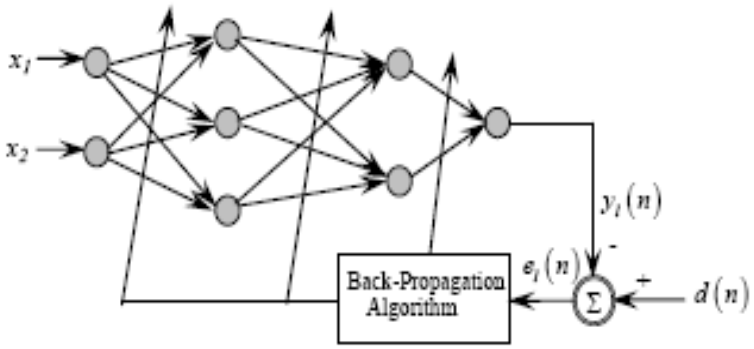


Fig. 3.3 Neural network using BP algorithm

An MLP network with 2-3-2-1 neurons (2, 3, 2 and 1 denote the number of neurons in the input layer, the first hidden layer, the second hidden layer and the output layer respectively) with the back-propagation (BP) learning algorithm, is depicted in Fig.3.3. The parameters of the neural network can be updated in both sequential and batch mode of operation. In BP algorithm, initially the weights and the thresholds are initialized as very small random values. The intermediate and the final outputs of the MLP are calculated by using (3.3), (3.4.), and (3.5.) respectively.

The final output $y_l(n)$ at the output of neuron *l*, is compared with the desired output *d(n)* and the resulting error signal $e_l(n)$ is obtained as

$$e_l(n) = d(n) - y_l(n) \tag{3.7}$$

The instantaneous value of the total error energy is obtained by summing all error signals over all neurons in the output layer, that is

$$\xi(n) = \frac{1}{2}\sum_{i=1}^{p_3} e_l^2\,(n) \tag{3.8}$$

where P_3 is the no. of neurons in the output layer.

This error signal is used to update the weights and thresholds of the hidden layers as well as the output layer. The reflected error components at each of the hidden layers is computed using the errors of the last layer and the connecting weights between the hidden and the last layer and error obtained at this stage is used to update the weights between the input and the hidden layer. The

thresholds are also updated in a similar manner as that of the corresponding connecting weights. The weights and the thresholds are updated in an iterative method until the error signal becomes minimum. For measuring the degree of matching, the Mean Square Error (MSE) is taken as a performance measurement. The updated weights are,

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n) \quad (3.9)$$

$$w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}(n) \quad (3.10)$$

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \quad (3.11)$$

where, $\Delta w_{kl}(n)$, $\Delta w_{jk}(n)$ *and* $\Delta w_{ij}(n)$ are the change in weights of the second hidden layer-to-output layer, first hidden layer-to-second hidden layer and input layer-to-first hidden layer respectively. That is,

$$\Delta w_{kl}(n) = -2\mu \frac{d\xi(n)}{dw_{kl}(n)} = \mu e(n) \frac{dy_l(n)}{dw_{kl}(n)}$$

$$= \mu e(n) \varphi' \left[\sum_{k=1}^{p_2} w_{kl} f_k + b_l \right] f_k \quad (3.12)$$

Where, μ is the convergence coefficient (0≤μ≤1)**.** Similarly the Δw_{jk} and Δw_{ij} can be computed The thresholds of each layer can be updated in a similar manner, i.e.

$$b_l(n+1) = b_l(n) + \Delta b_l(n) \quad (3.13)$$

$$b_k(n+1) = b_k(n) + \Delta b_k(n) \quad (3.14)$$

$$b_j(n+1) = b_j(n) + \Delta b_j(n) \quad (3.15)$$

where, $\Delta b_l(n), \Delta b_k(n), \Delta b_j(n)$ are the change in thresholds of the output, hidden and input layer respectively. The change in threshold is represented as,

$$\Delta b_l(n) = -2\mu \frac{d\xi(n)}{db_l(n)} = \mu e(n) \frac{dy_l(n)}{db_l(n)} = \mu e(n) \varphi' \left[\sum_{k=1}^{p_2} w_{kl} f_k + b_l \right] f_k \quad (3.16)$$

Chapter 4

RADIAL BASIS FUNCTIONS NETWORK

4. RADIAL BASIS FUNCTIONS NETWORK

4.1. INTRODUCTION

Radial Basis Function Networks (RBFN) are multilayer feed-forward neural networks consisting of one input layer, one hidden layer and one output layer with linear weights as shown in Fig4.1. The function of the hidden layer is to perform a non-linear transformation of the input space. The hidden layer typically comprises an activation function which is a non-linear function of the distance between the input space and the corresponding centers decided by the hidden space or rather, the Euclidean Norm of the input points and the centers. These activation functions which are real valued with values depending upon the radial distance of a point from the origin or center are called Radial Basis Functions and the Networks using them are hence called Radial Basis Function Networks(RBFNs). The hidden space is typically of higher dimensionality than the input space corresponding to Cover's theorem('65) which states that a complicated pattern classification problem that is non-linearly separable is more likely to be linearly classified if it is cast into a high dimensional space rather than a low dimensional one. The output layer that contains linear weights perform a linear regression to predict the desired targets. The structure is drawn from biological receptive fields to perform function mappings. Weights on the output layer are adapted via supervised learning.

4.2. RBFNN SRUCTURE

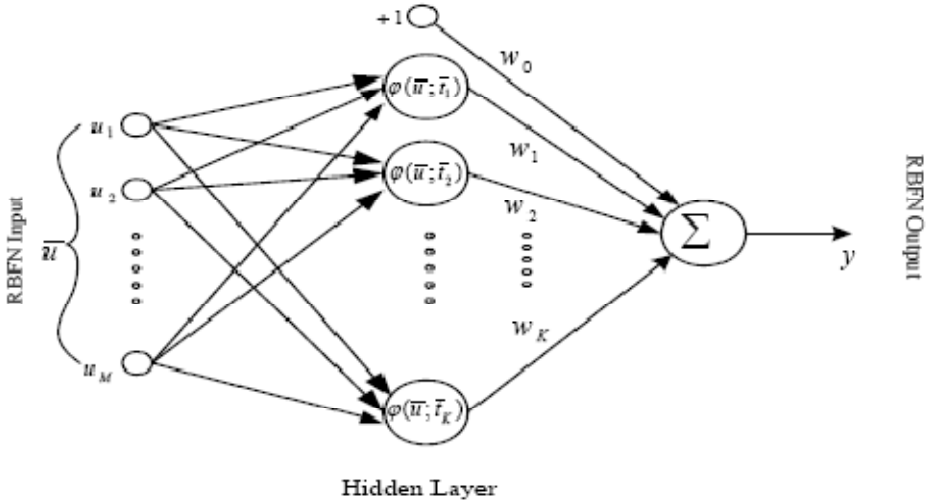


Fig.4.1 Structure of RBFNN

As shown in the figure, the input vector $\bar{u}$ of dimension *M* is in the input layer. Hidden layer contains the Radial Basis Functions that perform the nonlinear mapping. There are *K* nodes , so its dimensionality is *K* such that *K>M*. Each node has a center vector $\overline{t_k}$. The ouput layer contains the linear weights $W = [w_0\ w_1\ .....\ w_K]^T$ that perform linear regression. The input-output mapping is given by the following equation:

$$y = \sum_{k=1}^{K} w_k \varphi(\bar{u}\,;\overline{t_k}) + w_0 \tag{4.1}$$

4.2.1. VARIOUS RADIAL BASIS FUNCTIONS

A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin, so that $\varphi(x) = \varphi(\|x\|)$; or alternatively on the distance from some other point *t*, called a *center*, so that $\varphi(x,t) = \varphi(\|x-t\|)$.

RBF types :

1. Multiquadric
   $\varphi(r) = (r^2 + \beta^2)^{\frac{1}{2}}$ for $\beta > 0$ and $r = (\|x-t\|)$ (4.2)
2. Inverse Multiquadric
   $\varphi(r) = (r^2 + \beta^2)^{-\frac{1}{2}}$ for $\beta > 0$ and $r = (\|x-t\|)$ (4.3)
3. Gaussian
   $\varphi(r) = exp(-r^2/\sigma^2)$ for $\sigma > 0$ and $r = (\|x-t\|)$ (4.4)

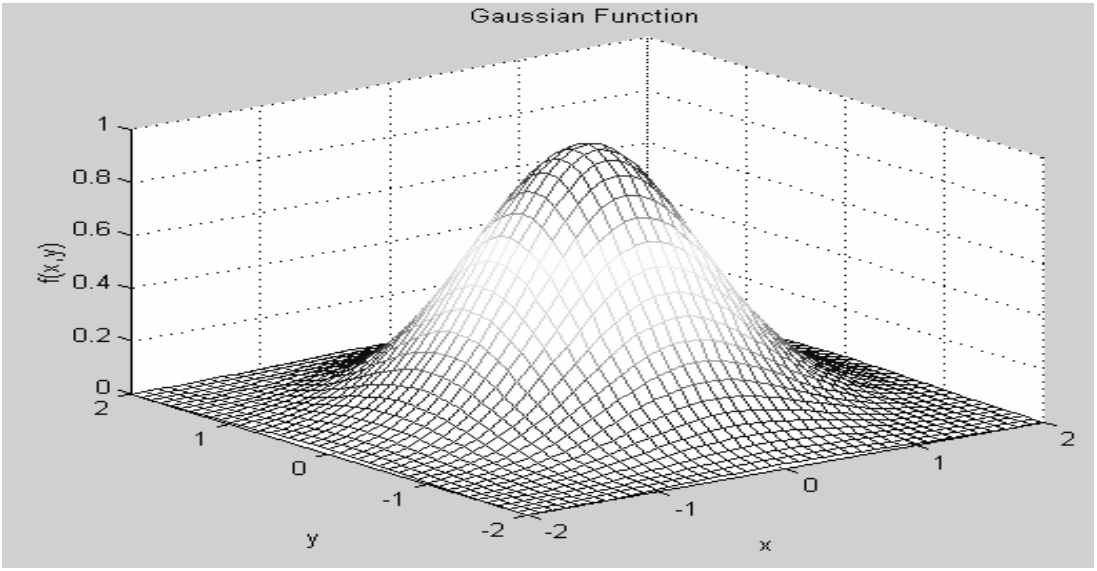


Fig.4.2 The Gaussian Function

In our context,

$$\varphi(\bar{u}\,;\overline{t_k}) = exp\left(-\frac{\left\|\bar{u}-\overline{t_k}\right\|^2}{\sigma^2}\right) \tag{4.5}$$

Where $\bar{u}$ is the input vector, $\overline{t_k}$ is the center vector σ_k is the width of the Gaussian function and $\left\|\bar{u}-\overline{t_k}\right\|$ is the Euclidean distance between $\bar{u}$ and $\overline{t_k}$. The Gaussian function is the most popular amongst the above.

As can be seen from the form of the radial basis functions , the multiquadric function increases monotonically with increase in *r* while the inverse multiquadric and Gaussian functions decrease with increase in *r*. Further, the rate at which the output decreases can be controlled by varying the width σ_k in case of a Gaussian kernel function. This means the output of the RBFN will decrease in case the input point is far from the centre and will tend to zero if we use Gaussian or inverse multiquadric functions and the output will increase if we use multiquadric. So ,theoretically speaking, RBFN with inverse multiquadric function is good for extrapolation whereas RBFN with inverse multiquadric or Gaussian functions is good for interpolation. It is note worthy that the most commonly used Radial Basis Function is the Gaussian function. So, we could safely say that RBFN is good for interpolation. It should be noted that we do not consider the Regularized Radial Basis Function Network which takes the same number of centers as the number of input points in the training set. This computationally very complex if we have even slightly large training sets. We rather would discuss in detail the Generalized Radial Basis Function Network(GRBFN) which has number of centers less than the number of input points. The number and location of these centers are chosen strategically so that function approximation and system identification problems can be solved with more precision and less computational complexity. Henceforth by RBFNN we would refer specifically to GRBFNN.

4.3. LEARNING STRATEGIES APPLIED TO GRBFNNs

Like a multilayer perceptron, RBFN has universal approximation ability. The advantages of RBFN are linearity in parameters and the availability of fast and efficient training methods. RBFN learns to approximate the desired input-output map represented by training data $\{\bar{u}_i, d_i\}$where $\bar{u}_i$ is the input vector and d_i is the desired response (target), *i = 1, 2,..., N*. A

number of learning methods exist to approximate the desired input-output maps. And by these learning methods we mean the efficient selection of the centers and a method to update the linear weights.

4.3.1. Fixed Centers Selected at Random

In this learning method, RBFs of the hidden units are fixed, that is, the centers are not updated ; they are fixed. The locations of the centers may be chosen randomly from the training data set. We can use different values of centers and widths for each radial basis function for which experimentation with training data is needed. Only the output layer weights need to be learned. The values of the output layer weights are obtained easily by pseudo-inverse method. This method is apparently very simple but to produce results that can show a satisfactory level of performance, it requires a large training set and rigorous experimentation on the training data.

4.3.2 Self-organized Selection of Centers

Self-organized selection of centers employs a hybrid learning approach which combines self-organized learning algorithm based on K-Means Clustering Algorithm and supervised learning algorithm based on stochastic gradient. The former is used to determine the center of Gaussian function, while the later is employed to adjust the output weights. The number of centers is depended on the number of clusters of data, or it could well be the user's discretion – an arbitrary selection.

K-means clustering algorithm is used to cluster data into *k* number of clusters. Specifically, this algorithm places centers of radial basis function in the input space area where the data are significant. K-means clustering algorithm proceeds as follows :

1. *Initialization*, select randomly center values $\overline{t_k}(0)$; the only requirement is that values of $\overline{t_k}(0)$ must be different for each $k = 1, 2, \ldots, K$. It is suggested that Euclidean norm of each center sufficiently small.
2. *Sampling*, take a sample vector *u* of input space with certain probability. Vector *u* represents input applied to RBFN.
3. *Similarity matching*, find center of the winner at n^{th} iteration, with minimum euclidean distance :

$$\tilde{k}(\bar{u}) = \arg min_k \left(\left\| \bar{u} - \overline{t_k} \right\|\right) \qquad k = 1, 2, \ldots, K \tag{4.6}$$

4. *Updating,* adjust center position according to:

$$\overline{t_k}(n+1) = \begin{cases} \overline{t_k}(n) + \eta\left[\bar{u} - \overline{t_k}\right], & k = \tilde{k}(\bar{u}) \\ \overline{t_k}(n), & otherwise \end{cases} \tag{4.7}$$

The spread or width of the Gaussian function is determined by taking $\sigma = \frac{d_{max}}{\sqrt{2K}}$ where d_{max} = maximum distance between the centers and *K*= number of nodes of RBFN.The weights are then updated by supervised learning using LMS.

4.3.3 Stochastic Gradient Approach (Supervised Learning)

In this method, RBF network design takes on its most generalized form. As we know, the RBFN has three parameters: centers $\overline{t_k}$, spread σ_k and output layer weights w_k. Here, all these parameters the centers $\overline{t_k}$, spread σ_k of the radial-basis functions and all the weights w_k of the network undergo a *supervised learning process.* A natural candidate is error-correction learning, using a stochastic gradient descent of the error criterion. The Basic concept of this method is similar to LMS algorithm.

Algorithm:

- We take the Cost Function $\xi(n) = \frac{1}{2}|e(n)|^2$ for n = 1,2,……..,N
- Where *e(n)* is the error signal
- $e(n) = d(n) - y(n)$

  $= d(n) - \sum_{k=1}^{K} w_k \varphi(\bar{u}; \overline{t_k})$

  $= d(n) - \sum_{k=1}^{K} w_k \, exp\left(-\left\|\bar{u} - \overline{t_k}\right\|^2/\sigma^2\right)$
- To minimize *ξ(n),* we would use the stochastic gradient descent method :

$$w_k(n+1) = w_k(n) + \eta_w \frac{\partial \xi(n)}{\partial w_k(n)}$$

$$\overline{t_k}(n+1) = \overline{t_k}(n) + \eta_t \frac{\partial \xi(n)}{\partial \overline{t_k}(n)}$$

$$\sigma_k(n+1) = \sigma_k(n) + \eta_\sigma \frac{\partial \xi(n)}{\partial \sigma_k(n)}$$

Hence the results of the stochastic gradient approach can be summarized as:

- $y(n) = \sum_{k=1}^{K} w_k \varphi(\, \bar{u}(n)\, ;\, \overline{t_k}(n)\,) + \, w_0$ (1)
- $e(n) = d(n) - y(n)$ (2)

Parameter Update Equations:

- $w_k(n+1) = \, w_k(n) + \, \eta_w \, e(n) \varphi(\, \bar{u}(n)\, ;\, \overline{t_k}(n)\,)$ (3)

- $\overline{t_k}(n+1) = \, \overline{t_k}(n) + \, \eta_t e(n) \, \varphi\big(\, \bar{u}(n)\, ;\, \overline{t_k}(n)\big) \, (\bar{u}(n) - \overline{t_k}(n))/\sigma_k^2(n)$ (4)

- $\sigma_k^2(n+1) = \, \sigma_k^2(n) + \, \eta_t e(n) \, \varphi\big(\, \bar{u}(n)\, ;\, \overline{t_k}(n)\big) \, (\left\| \, \bar{u} - \overline{t_k} \right\|^2)/\sigma_k^2(n)$ (5)

  where $\varphi\big(\, \bar{u}\, ;\, \overline{t_k}\,\big) = \, exp\left[- \frac{\left\| \, \bar{u} - \overline{t_k} \right\|^2}{\sigma^2} \right]$ (6)

Chapter 5

WILCOXON LEARNING MACHINES

5. WILCOXON LEARNING MACHINES

5.1 INTRODUCTION

Machine learning, namely learning from examples, has been an active research area for several decades. Popular and powerful learning machines proposed in the past include artificial neural networks , generalized radial basis function networks (GRBFNs) ,fuzzy neural networks (FNNs) and support vector machines (SVMs). They are different in their origins, network configurations, and objective functions. They have also been successfully applied in many branches of science and engineering. In statistical terms, the aforementioned learning machines are nonparametric in the sense that they do not make any assumptions of the functional form, e.g., linearity, of the discriminant or predictive functions. A detailed discussion of two of the above machines has been done, namely ANNs and GRBNs.

Robust smoothing is a central idea in statistics that aims to simultaneously estimate and model the underlying structure. In statistics, an outlier is an observation that is numerically distant from the rest of the data. Hence, outliers are data points that are not typical of the rest of the data. Statistics derived from data sets that include outliers may be misleading. Depending on their location, outliers may have moderate to severe effects on the regression model. A regressor or a learning machine is said to be robust if it is not sensitive to outliers in the data.

As is well known in statistics, the resulting linear regressors by using the rank-based Wilcoxon approach to linear regression problems are usually robust against (or insensitive to) outliers. It is then natural to generalize the Wilcoxon approach for linear regression problems to nonparametric Wilcoxon learning machines for nonlinear regression problems. The prime motivation behind this thesis is to apply and study the Wilcoxon approach to the machines we studied before (ANN and GRBFN) and see how these machines perform in presence of outliers. We would try to demonstrate that these Wilcoxon learning machines are robust against outliers.

5.2 WILCOXON NORM

Before investigating the Wilcoxon learning machines, an introduction to the Wilcoxon Norm is required. To define the Wilcoxon norm of a vector, we need a score function. A score function is a function $\varphi(u)$: $[0,1] \longrightarrow \boldsymbol{R}$ which is non-decreasing such that

$$\int_0^1 \varphi^2(u)du < \infty$$

Usually the score function is standardized such that

$$\int_0^1 \varphi^2(u)du = 1 \quad and \quad \int_0^1 \varphi(u)du = 0$$

The score associated with the score function φ is defined by

$$a(i) = \varphi\left(\frac{i}{l+1}\right)$$

Where l is a fixed positive number.

It can be shown that the following function is a pseudonorm (seminorm) on $\boldsymbol{R}^l$,

$$\|v\|_W := \sum_{i=0}^{l} a[R(v_i)].v_i = \sum_{i=0}^{l} a(i)v_{(i)}$$
$$v := [v_0\, v_1 \ldots\ldots v_l]^T \tag{5.1}$$

Where $R(v_i)$ is the rank of v_i among $v_0, v_1 \ldots\ldots v_l$. $v_{(0)} \leq v_{(1)} \ldots\ldots \leq v_{(l)}$ are the ordered values of $v_0\, v_1 \ldots\ldots v_l$. $a(i) = \varphi\left(\frac{i}{l+1}\right)$ and $\varphi(u) = \sqrt{12}\,(u - 0.5)$.

$\|v\|_W$ is called the Wilcoxon Norm of the vector v.

5.3 WILCOXON NEURAL NETWORK WNN

5.3.1 Neural Network Structure

We consider a three layered neural network with one input , one hidden and one output layer. This neural network is for the analysis of a general input-output mapping of n dimensions to p dimensions, i.e. input vector o n dimensions is to be mapped to an output of p dimensions. Hence we consider the following network of $n+1$ input nodes $m+1$ hidden nodes and p output nodes.

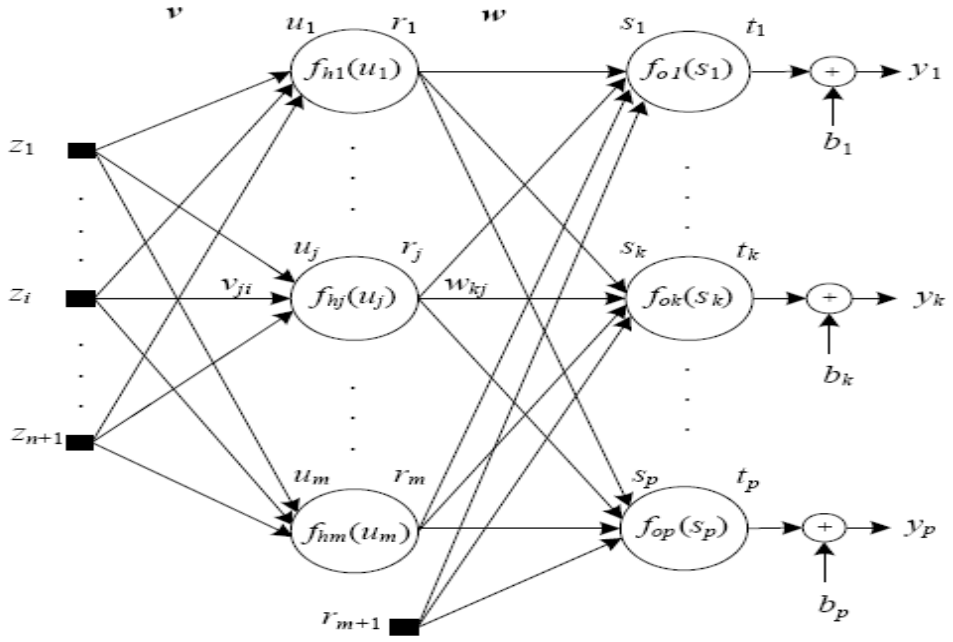


Fig.5.1 Wilcoxon Neural Network Structure

Let the input vector be $x := [x_1\ x_2\ \ldots\ldots x_n]^T$ or $z := [z_1\ z_2\ \ldots\ldots z_n\ z_{n+1}]^T = [x_1\ x_2\ \ldots\ldots x_n\ 1]^T$

Let v_{ji} denote the connection weight from the *i*th input node to the input of the *j*th hidden node. Then, the input u_j and output r_j of the *j*th hidden node are given by, respectively

$$u_j := \sum_{i=1}^{n+1} v_{ji}.z_i \quad , \quad z_{n+1} = 1 \ , \qquad r_j = f_h(u_j) \qquad j \in m \tag{5.2}$$

where $f_h(.)$ is the activation function of the hidden node. Some commonly used activation functions are sigmoidal functions, i.e., monotonically increasing S-shaped functions as follows:

- Unipolar logistic function $\qquad r_j = f_h(u_j) = \frac{1}{1+e^{-u_j}}$
- Bipolar sigmoidal function $\qquad r_j = f_h(u_j) = \frac{1-e^{-u_j}}{1+e^{-u_j}}$
- Hyperbolic tangent function $\qquad r_j = f_h(u_j) = \frac{e^{-u_j}-e^{-u_j}}{e^{-u_j}+e^{-u_j}}$

Let w_{kj} denote the connection weight from the output of the *j*th hidden node to the input of the *k*th output node. Then, the input u_j and output r_j of the *j*th hidden node are given by, respectively

$$s_k := \sum_{i=1}^{m+1} w_{kj}.r_j \quad , \quad r_{m+1} = 1 \ , \qquad t_k = f_o(s_k) \qquad k \epsilon p$$

(5.3)

where $f_o(.)$ is the activation function of the output node. For classification problems, the output activation functions can be chosen as sigmoidal functions, while for regression problems, the output activation functions can be chosen as linear functions with unit slope.

The final output y_k of the network is given by $y_k = t_k + b_k \quad , \quad k \epsilon p$

Where b_k is the bias.

We define

$$u := [u_1 \ldots\ldots u_m]^T \quad \epsilon \ R^m$$
$$r := [r_1 \ldots\ldots r_m \, r_{m+1}]^T \ \epsilon \ R^{m+1} \qquad r_{m+1} = 1$$
$$w_k := \left[w_{k1} \ldots\ldots w_{km} \, w_{k(m+1)}\right]^T \ \epsilon \ R^{m+1}$$

(5.4a)

$$V := \left[v_{ji}\right]^T \quad \epsilon \ R^{mX(n+1)} \tag{5.4b}$$

From 5.2 – 5.4 we have,

$$u = Vz \ ; \quad r = f_h(u) \ ; \quad s_k = {w_k}^T r \ ; \quad t_k = f_o(s_k) \ ; \quad y_k = t_k + b_k \quad k \epsilon p \tag{5.5}$$

Let $X \subseteq R^n \ and \ Y \subseteq R^p$. We are given a training set

$$S := \{(x_q, y_q)\}_{q=1}^{l} \subseteq X \times Y$$
$$S := \{(z_q, y_q)\}_{q=1}^{l} \subseteq R^{n+1} \times R^p$$

In the following, we will use the subscript to denote the qth example.

In a WNN, the approach is to choose network weights that minimize the Wilcoxon norm of the total residuals

$$\Psi_{\text{total}} = \sum_{k=1}^{p} \sum_{q=1}^{l} a\left[R(\rho_{qk})\right].\rho_{(q)k} = \sum_{k=1}^{p} \sum_{i=0}^{l} a(q)\rho_{(q)k}$$

(5.6a)

$$\rho_{qk} = d_{qk} - t_{qk} \qquad q \epsilon l \qquad k \epsilon p \tag{5.6b}$$

Where $R(\rho_{qk})$ is the rank of ρ_{qk} among $\rho_{0k}, \rho_{1k} \ldots\ldots \rho_{lk}$. $\rho_{(0)k} \leq \rho_{(1)k} \ldots\ldots \leq \rho_{(l)k}$ are the ordered values of $\rho_{0k}, \rho_{1k} \ldots\ldots \rho_{lk}$ .the Wilcoxon norm of residuals at the kth output node is given by

$$\Psi_{\mathrm{k}} := \|\rho_q\|_W := \sum_{q=1}^{l} a[R(\rho_{qk})].\rho_{(q)k} = \sum_{i=0}^{l} a(q)\rho_{(q)k} \tag{5.7a}$$

$$\rho_{qk} = d_{qk} - t_{qk} \qquad q \in l \qquad k \in p \tag{5.7b}$$

$$\rho_k := [\rho_{1k} \ldots\ldots \rho_{lk}]^T \in R^l \tag{5.7c}$$

$$\Psi_{\text{total}} = \sum_{k=1}^{p} \Psi_{\mathrm{k}}$$

The NN used here is the same as that used in standard ANN, except the bias terms at the outputs. The main reason is that the Wilcoxon norm is not a usual norm, but a pseudonorm (seminorm).In particular $\|v\|_W = 0$ for $v := [v_0\ v_1 \ldots\ldots v_l]^T$ implies that $v_0 = v_1 = \cdots = v_l$. This means that, without the bias terms, the resulting predictive function with small Wilcoxon norm of total residuals may deviate from the true function by constant offsets.

5.3.2. Learning Algorithm of WNN

Now, we introduce an incremental gradient–descent algorithm. In this algorithm, Ψ_{k}s are minimized in sequence. From the definition of Ψ_{k} in (5.7) together with (5.5), we have

$$\begin{aligned}
\Psi_{\mathrm{k}} := \|\rho_q\|_W &:= \sum_{q=1}^{l} a[R(\rho_{qk})].\rho_{(q)k} = \sum_{i=0}^{l} a(q)\rho_{(q)k} \\
&= \sum_{i=0}^{l} a(q)(\, d_{(q)k} - t_{(q)k}\,) \\
&= \sum_{i=0}^{l} a(q)(\, d_{(q)k} - f_o(s_{(q)k})\,) \\
&= \sum_{i=0}^{l} a(q)(\, d_{(q)k} - f_o(w_k{}^T r_{(q)})\,) \\
&= \sum_{i=0}^{l} a(q)(\, d_{(q)k} - f_o\left(w_k{}^T f_h(u_{(q)})\right)\,) \\
&= \sum_{i=0}^{l} a(q)\left(\, d_{(q)k} - f_o\left(w_k{}^T f_h(Vz_{(q)})\right)\,\right)
\end{aligned}$$

Updating rules for the weights connecting input layer to hidden layer and those connecting hidden to output layer.

$$w_k \leftarrow w_k - \eta \frac{\partial \Psi_k}{\partial w_k}$$

$$V \leftarrow V - \eta \frac{\partial \Psi_k}{\partial V}$$

$\eta > 0$ is the learning rate.

From (5.8) we have,

$$\frac{\partial \Psi_k}{\partial w_k} = \sum_{i=0}^{l} a(q)(-1) f'_o(s_{(q)k})\, r_{(q)}$$

$$= \sum_{i=0}^{l} a[R(\rho_{qk})](-1) f'_o(s_{qk})\, r_q$$

where $f'_o(.)$ denotes the total derivative of $f_o(.)$with respect to its argument and s_{qk} is the *k*th component of the *q*th vector . Hence, the updating rule becomes

$$w_k \leftarrow w_k + \eta \frac{\partial \Psi_k}{\partial w_k} = w_k + \eta \sum_{i=0}^{l} a[R(\rho_{qk})](-1) f'_o(s_{qk})\, r_q \qquad k \in p$$

i.e.,

$$w_{kj} \leftarrow w_{kj} + \eta \sum_{i=0}^{l} a[R(\rho_{qk})] f'_o(s_{qk})\, r_{qj} \tag{5.9}$$

Again,

$$\frac{\partial \Psi_k}{\partial V} = \sum_{i=0}^{l} a(q)(-1) f'_o(s_{(q)k}) \begin{bmatrix} w_{k1} f'_h(u_1) \\ w_{k2} f'_h(u_2) \\ \dots \\ w_{km} f'_h(u_m) \end{bmatrix}_{(q)} [z_1 \dots . z_n\ z_{n+1}]_{(q)}$$

Hence the updating rule becomes

$$v_{ji} \leftarrow v_{ji} + \eta \sum_{i=0}^{l} a[R(\rho_{qk})] f'_o(s_{qk})\, w_{kj}\, f'_h(u_{qj})\, z_{qi} \qquad j \in m \ \text{and}\ i \in n+1 \tag{5.10}$$

where $f'_h(.)$ denotes the total derivative of $f_h(.)$with respect to its argument and u_{qj} is the *j*th component of the *q*th vector .

The bias term $b_k\ , k\ \epsilon\ p$ is given by the median of the residuals at the *k*th output node i.e.

$$b_k = \underset{1 \le q \le l}{median}\{d_{qk} - t_{qk}\} \tag{5.11}$$

5.4 WILCOXON GENERALISED RADIAL BASIS FUNCTION NETWORK (WGRBFN)

The Wilcoxon approach to GRBFN is similar to the approach used in ANN. In fact, the three layer network we considered in fig5.1 can be conceptualized as a GRBFN if we replace the activation function of the hidden layer by the Gaussian function used in RBF and taking the output layer activation function as a linear function with unity slope.

Continuing our treatment using fig5.1

We define $x\ :=\ [x_1\ x_2\ \ldots..x_n]^T\ \epsilon\ R^n$ and $y\ :=\ \left[y_1\ y_2\ \ldots..y_p\right]^T\ \epsilon\ R^p$

The predictive function f is a non-linear map given by

$$y_k = f_h(x) = \sum_{j=1}^{m} w_{kj} exp\left(-\sum_{i=1}^{n} \frac{\left\| x_i - t_{ji}\right\|^2}{2\sigma_{ji}^2}\right) +\ b_k\ , k\ \epsilon\ p \tag{5.12}$$

Here, w_{kj} is the connection weight between *j*th hidden node to *k*th output. $t_j\ :=\ \left[t_{j1}, t_{j1}\ \ldots..t_{jn}\right]^T$ is thecenter of the *j*th basis function. $2\sigma_{ji}^2$ is the *i*th variance of the *j*th basis function and b_k is the bias term. This system can also be represented as a feed-forward network. In this network, there are one input layer with nodes, one hidden layer with nodes, and one output layer with nodes. We also have bias terms at the output nodes.

Defining for $i\ \epsilon\ n\ , j\ \epsilon\ p\ , k\ \epsilon\ p$

$$u_j = -\sum_{i=1}^{n} \frac{\left\| x_i - t_{ji}\right\|^2}{2\sigma_{ji}^2}\ ,\qquad r_j = \exp(u_j)\ ,\ \ t_k = \sum_{i=0}^{l} w_{kj}\, r_j$$

Then from 5.12 we have

$$y_k\ =\ t_k +\ b_k$$

Suppose we are given the same training set as in Section 5.3 The Wilcoxon norm Ψ_k of residuals at the *k*th output node is the same as defined in Section 5.3. The incremental gradient–descent algorithm requires that Ψ_ks be minimized in sequence. By similar derivations, the weights updating rules are given by

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial \Psi_k}{\partial w_{kj}} = w_{kj} + \eta \sum_{i=0}^{l} a\left[R(\rho_{qk})\right] r_{qj}$$

$$t_{ji} \leftarrow t_{ji} - \eta \frac{\partial \Psi_k}{\partial t_{ji}} = t_{ji} + 2\eta \sum_{i=0}^{l} a\left[R(\rho_{qk})\right] r_{qj} \frac{(x_i - t_{ji})}{2\sigma_{ji}^2}$$

$$\sigma_{ji}^2 \leftarrow \sigma_{ji}^2 - \eta \frac{\partial \Psi_k}{\partial \sigma_{ji}^2} = \sigma_{ji}^2 + 2\eta \sum_{i=0}^{l} a\left[R(\rho_{qk})\right] r_{qj} \frac{\left\| x_i - t_{ji} \right\|^2}{2\sigma_{ji}^2} \tag{5.13}$$

Where $\eta > 0$ is the learning rate and the bias term b_k , $k \in p$ is given by the median of the residuals at the *k*th output node i.e.

$$b_k = \underset{1 \le q \le l}{median} \{d_{qk} - t_{qk}\}$$

Chapter 6

SIMULATIONS & CONCLUSION

6.1 SIMULATIONS

In this section, we compare the performances of various learning machines for several illustrative nonlinear regression problems. Emphasis is put particularly on the robustness against outliers for various learning machines. The updating rules for WNN are (9) and (10), WGRBFN are (13). It should be pointed out that different parameter settings for learning machines might produce different results. The parameters of each learning machine used in the following simulation may not be the optimal parameters for a given learning problem. This is the model selection problem, which always exists for a general learning problem. For "fair" comparison, similar machines will use the same set of parameters in the simulation. Thus, for ANN and WNN, we use the same number of hidden nodes, the same activation functions for hidden nodes, and the output node. Similarly, for GRBFN and WGRBFN, we use the same kernel function for both machines

In each simulation of Examples 1 and 2, the uncorrupted training data set consists of 50 randomly chosen *x*-points(training patterns) with the corresponding *y*-values (target values) evaluated from the underlying true function. The corrupted training data set is composed of the same -points as the corresponding uncorrupted one but with randomly chosen -values corrupted by adding random values from a uniform distribution defined on [-1,1]. It would be interesting to know what happens if the noise is progressively increased and if the number of outliers is increased. To this end, 20%, 30%, and 40% randomly chosen *y*-values of the training data points will be corrupted.

PERFORMANCE COMPARISION OF ANN & WNN

Example 1 :

$$y = \begin{cases} 1, & x = 0 \\ \dfrac{\sin x}{x}, & x \neq 0 \end{cases} \qquad x = [-10,10]$$

In this example, we compare the performances of ANN, WNN. For ANN and WNN, the number of hidden nodes is 30, the activation functions of the hidden nodes are bipolar sigmoidal functions, and the activation function of the output node is a linear function with unit slope. As we can see the input and output both are one dimensional so here referring to fig.5.1 , n=1,m=30,p=1. The results are plotted in the figures that follow.

All the figures have input values "x" on the *x*-axis and the corresponding estimates on the *y*-axis.

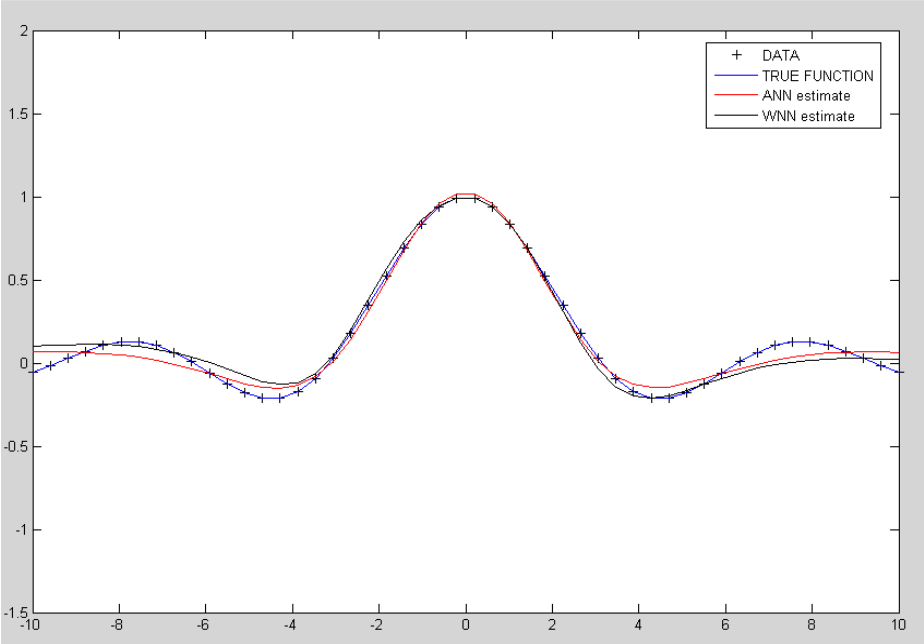


Fig.6.1 Performance of ANN & WNN -uncorrupted data

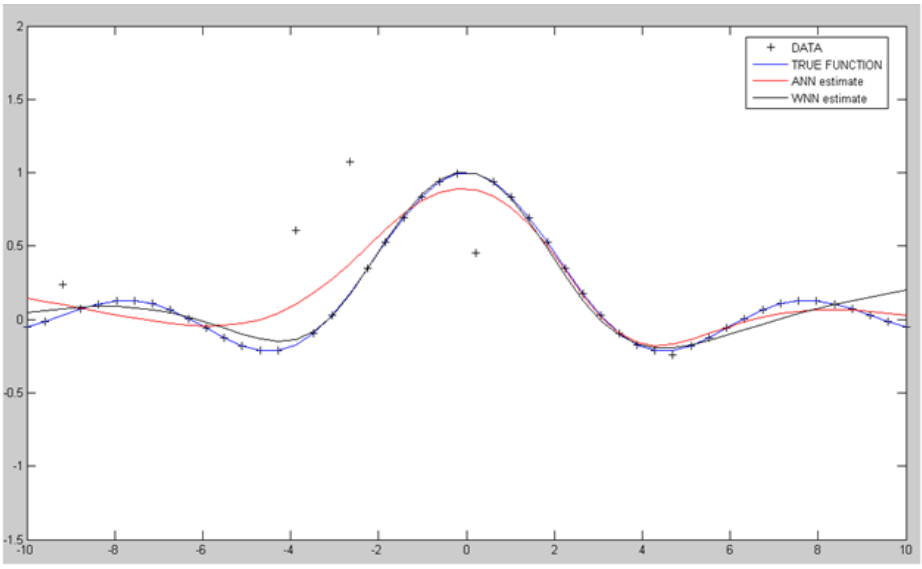


Fig.6.2 Performance of ANN & WNN -10% corrupted data

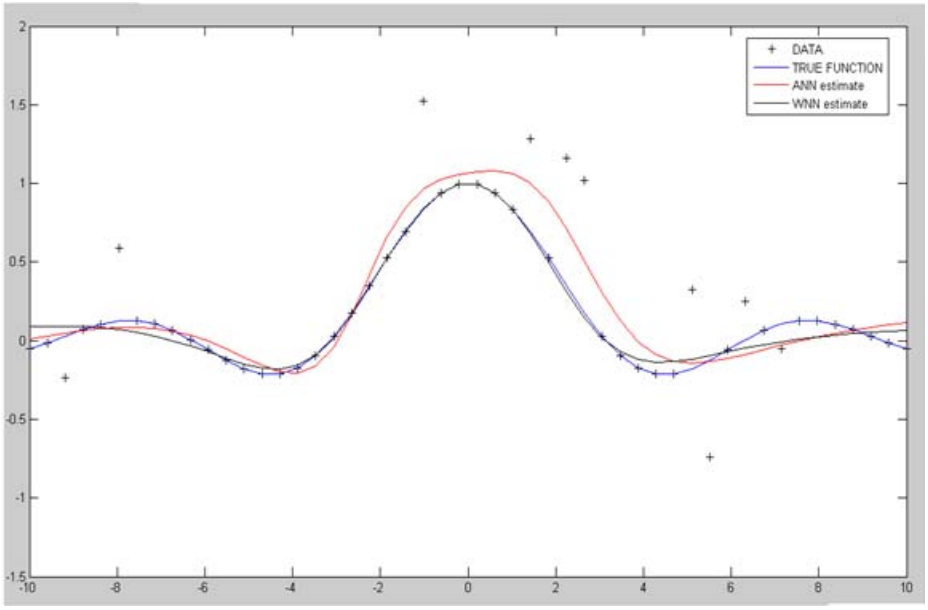


Fig.6.3 Performance of ANN & WNN -20% corrupted data

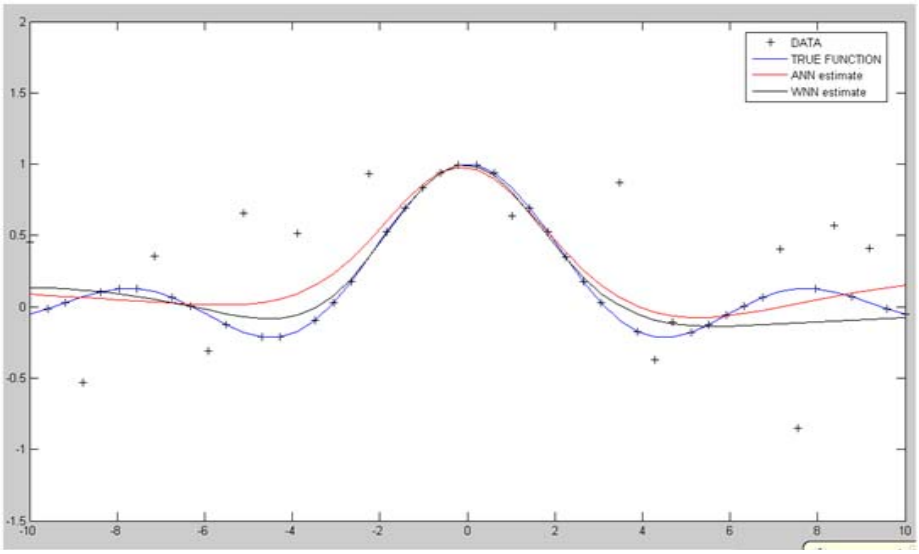


Fig.6.4 Performance of ANN & WNN -30% corrupted data

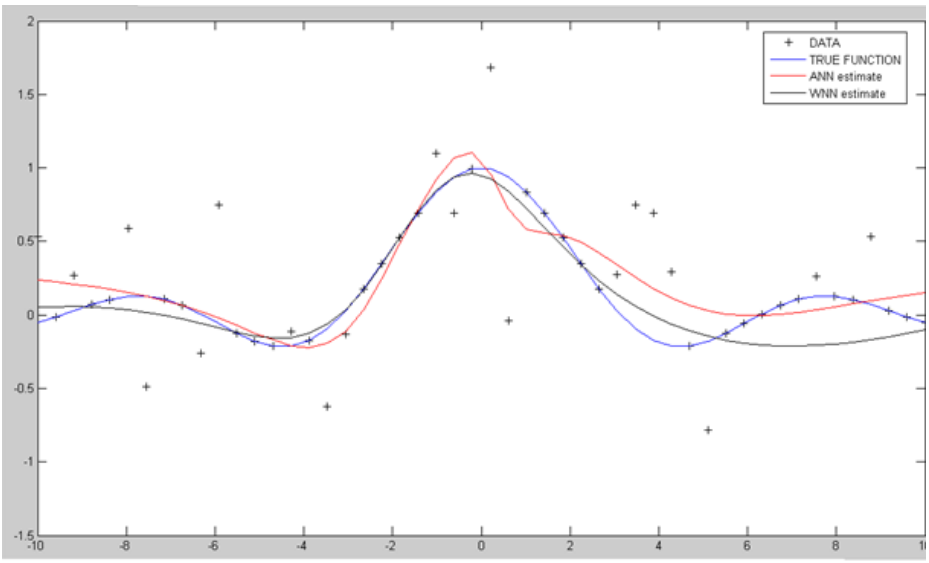


Fig.6.5 Performance of ANN & WNN -40% corrupted data

Example 2:

$$y = 1.1\,.\,(1 - x + x^2)\,e^{-x^2} \quad , \quad x = [-5{,}5]$$

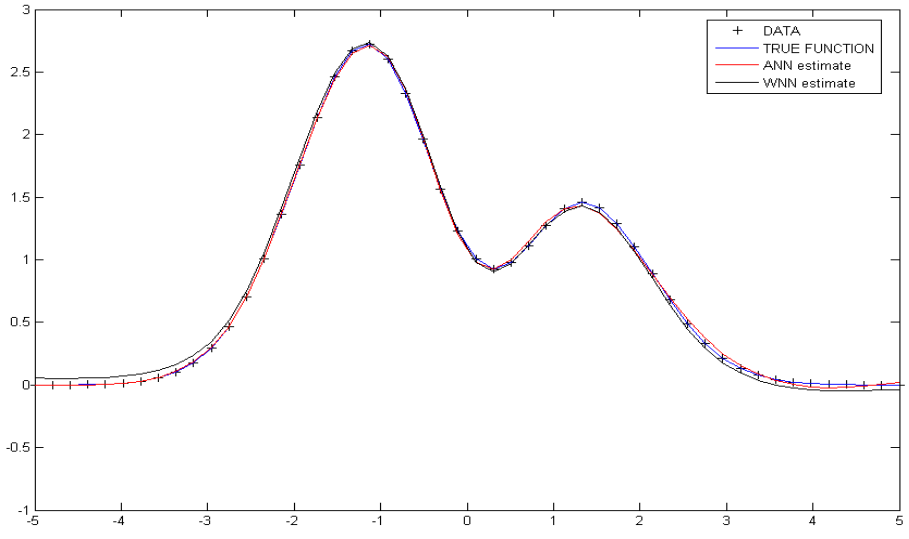


Fig.6.6 Performance of ANN & WNN -uncorrupted data

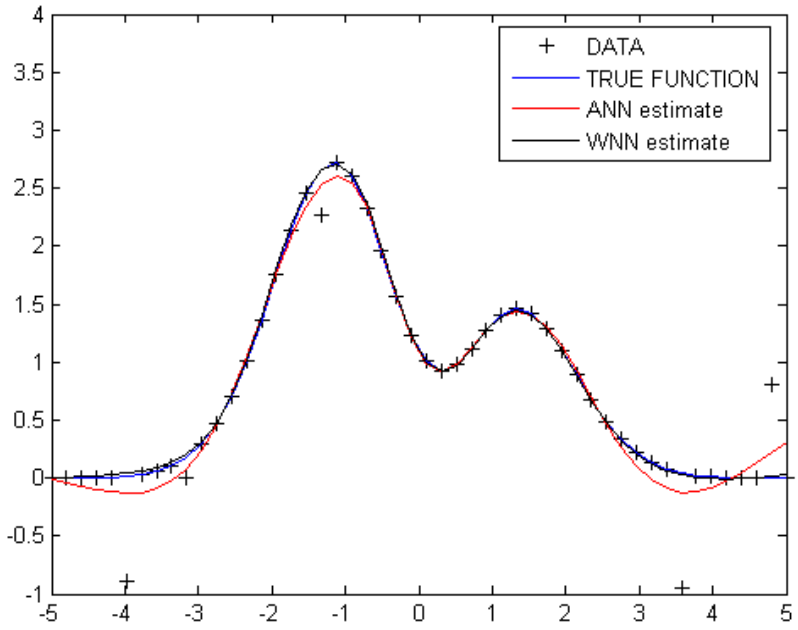


Fig.6.7 Performance of ANN & WNN -10% corrupted data

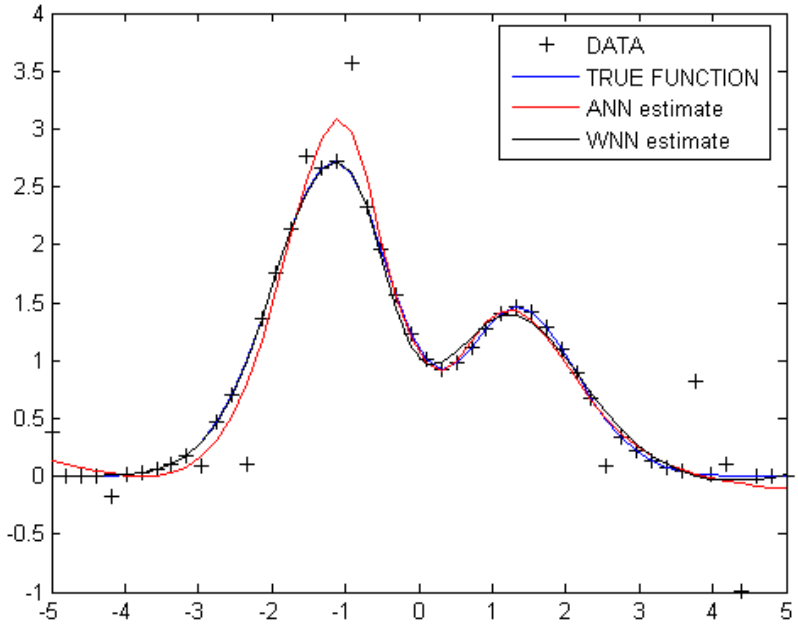


Fig.6.8 Performance of ANN & WNN -20% corrupted data

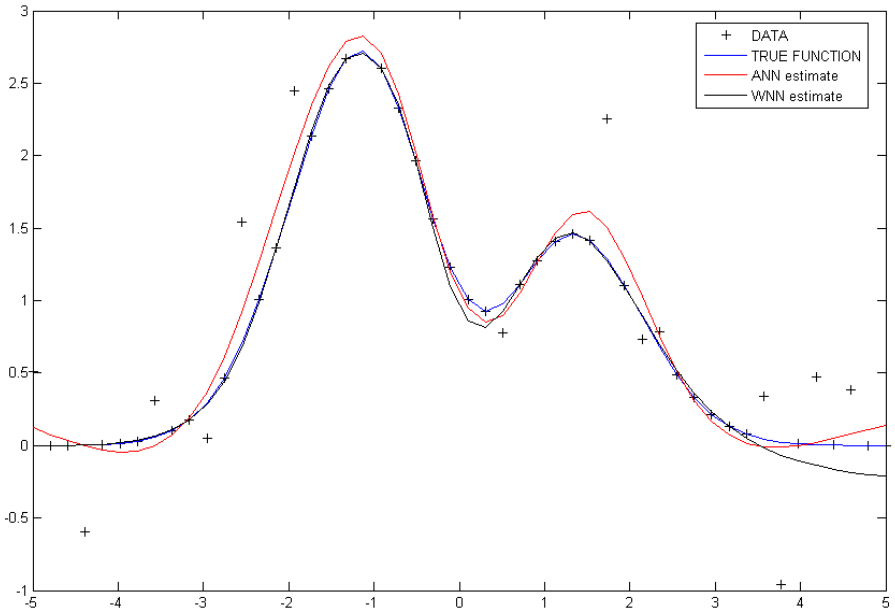


Fig.6.9 Performance of ANN & WNN -30% corrupted data

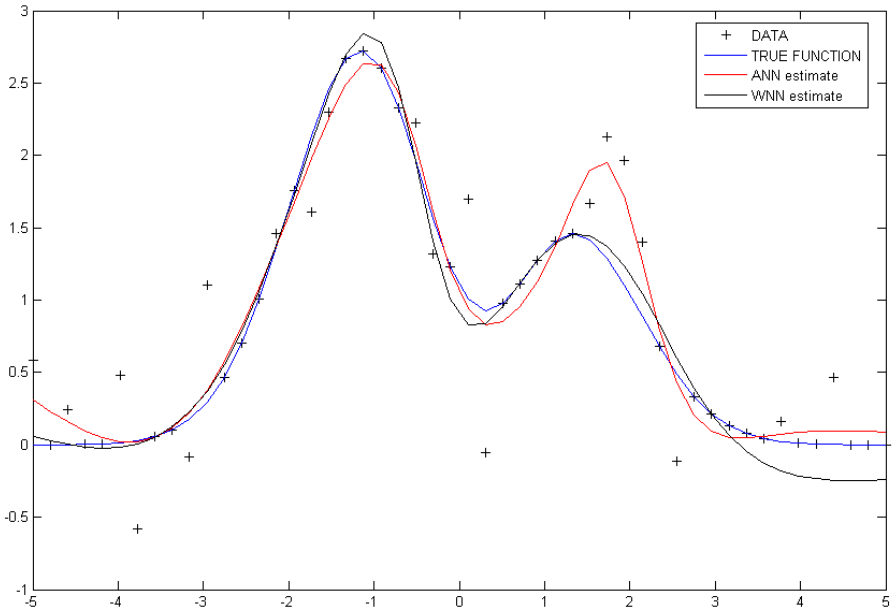


Fig.6.10 Performance of ANN & WNN -40% corrupted data

PERFORMANCE COMPARISION OF GRBFN & WGRBFN

Example 2: The true function is given by the Hermite function

$$y = 1.1\,.\,(1 - x + x^2)\,e^{-x^2} \quad , \quad x = [-5,5]$$

In this example, we compare the performances of GRBFN & WGRBFN. For all these networks, the number of hidden nodes is 20, which is somewhat arbitrary. The range for the training targets is [0.0002,2.7157] . The simulation results for GRBFN and WGRBFN are shown in the following figures.

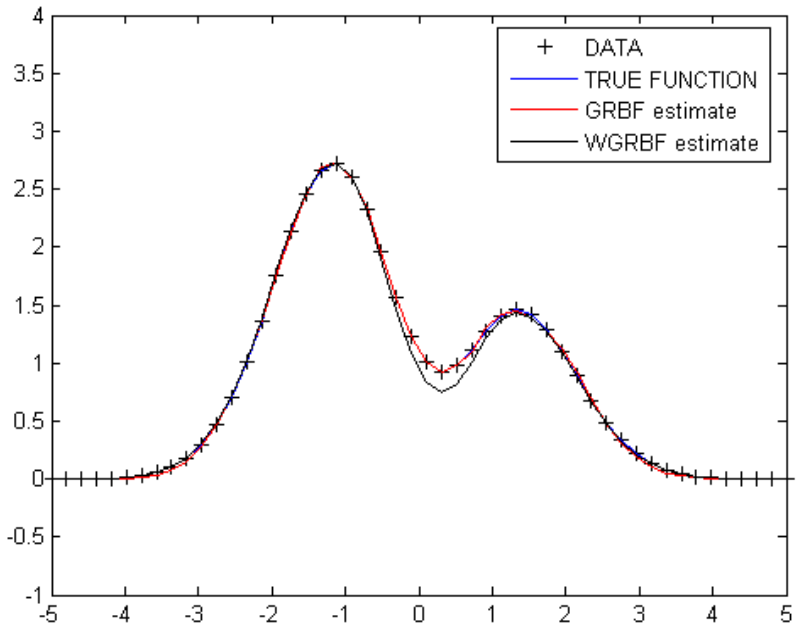


Fig.6.11 Performance of GRBFN & WGRBFN -uncorrupted data

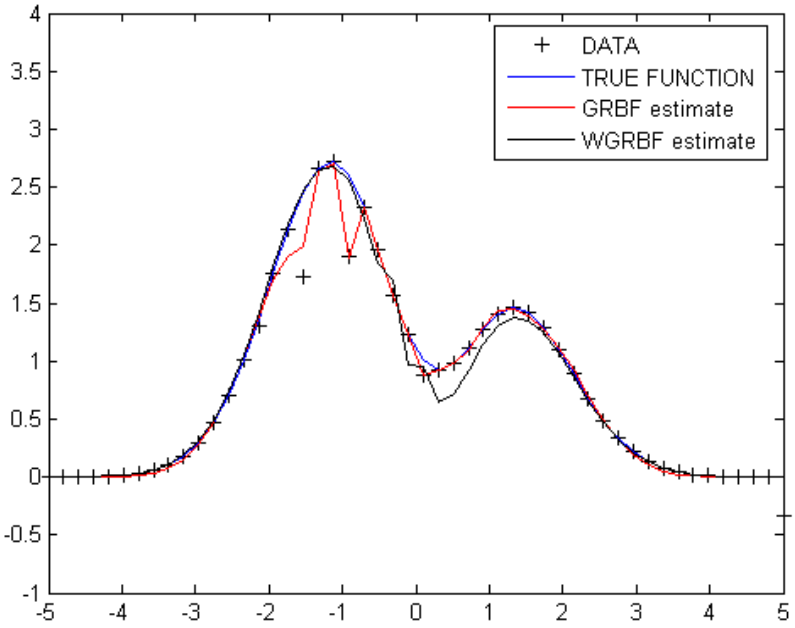


Fig.6.12 Performance of GRBFN & WGRBFN -10%corrupted data

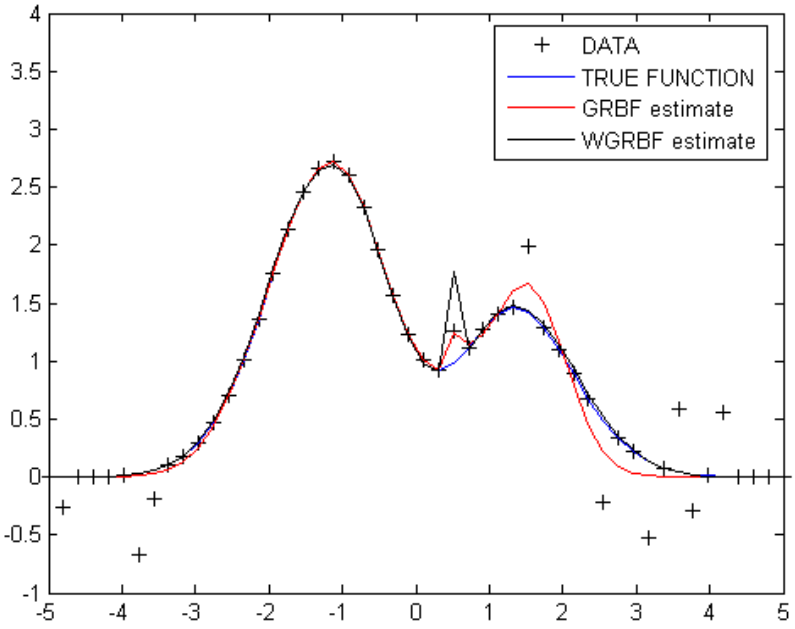


Fig.6.13 Performance of GRBFN & WGRBFN -20%corrupted data

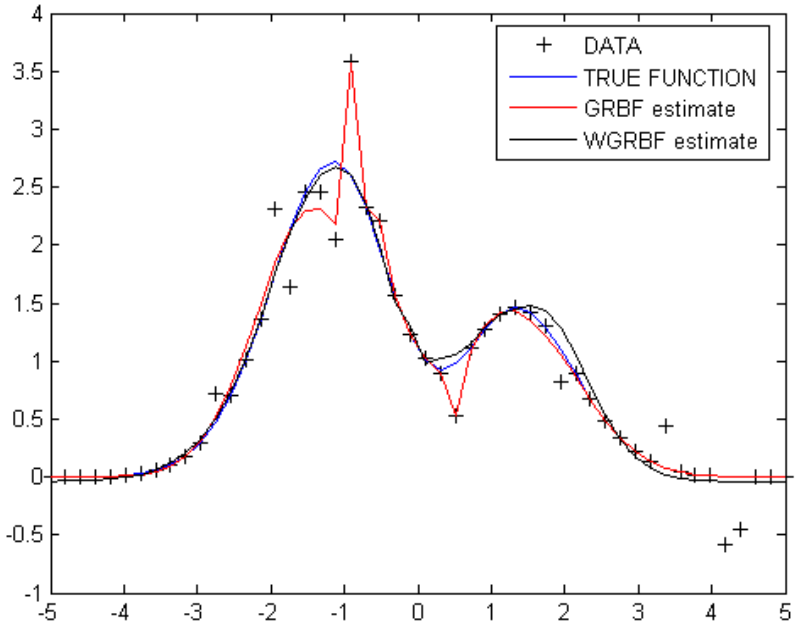


Fig.6.14 Performance of GRBFN & WGRBFN -30%corrupted data

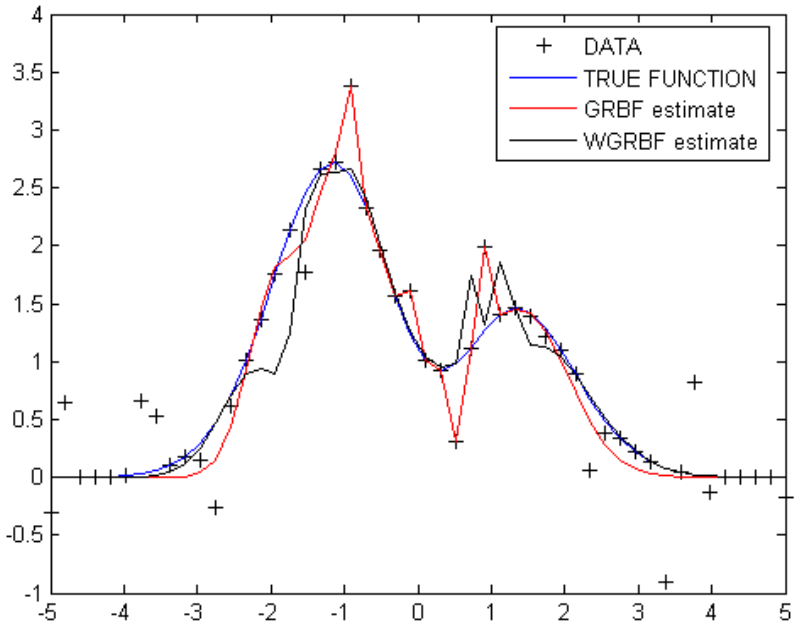


Fig.6.15 Performance of GRBFN & WGRBFN -40%corrupted data

6.2 CONCLUSION

The simulation results for ANN and WNN are shown in Fig. 6.1-6.10.Two examples are taken. The range for the training targets is [0.2171, 0.99879] in the first example and [0.0002, 2.7157] in the second one. For uncorrupted data shown in Fig. 6.1 & 6.6, WNN performs better than ANN and we are not over fitting the training data. For corrupted data shown in Fig. 6.2–6.5 and Fig. 6.6–6.10 with progressively increased corruption, WNN estimates are affected to a lot lesser extent by these corrupted outliers and outperform ANN estimates.

In case of simulations of GRBFN and WGRBFN, we take only one example of the function to be approximated. We take the Hermite function. For uncorrupted data shown in Fig. 6.11, GRBFN and WGRBFN estimates are almost indistinguishable from the true function and we are not over fitting the training data. For corrupted data shown in Fig. 6.12–6.15 with progressively increased corruption, WGRBFN estimates are robust to outliers since they are affected to a lot lesser extent by these corrupted outliers and they outperform GRBFN estimate.

This thesis demonstrates the Wilcoxon approach to nonlinear learning problems for the ANNs and GRBFNs. These provide alternative learning machines when faced with general nonlinear learning problems. Simple weights updating rules based on gradient descent were derived. Some numerical examples were provided to compare the robustness against outliers for standard learning machines and Wilcoxon learning machines. Simulation results showed that the Wilcoxon learning machines have good robustness against outliers.

The computational performances of the Wilcoxon learning machines are not discussed in this study. The reason is that it is still very time-consuming to obtain the numerical solutions of the Wilcoxon learning problems so that it makes little sense at this moment to present the data for computational performances of the Wilcoxon learning machines. The search of more efficient learning rules for Wilcoxon learning machines could be a future prospect. We are in the process of developing a novel learning machine based on FLANN using the Wilcoxon approach. The simulations of this machine are not ready yet. There was also an attempt on our side to develop algorithms based on LMS using Wilcoxon approach (we might call it WLMS) for linear

regression problems. The reason it hasn't been introduced in this thesis is for brevity and that the algorithm is still very computationally expensive.

It is true illustrative examples do not provide a rigorous proof for the robustness of the Wilcoxon learning machines. The results reported in this thesis provide only a start up or just a preliminary study on Wilcoxon learning machines. Similar approach can be used to other learning machines. As a final thought, we could only say that it is just a matter of time when we could actually see the application of Wilcoxon Norms and possibly other novel methodologies being developed for outlier rejection and robustness. In literature, much has been written on increasing the robustness of various learning machines against outliers. Wilcoxon Learning Machines could well be the answer to this very old problem.

6.3 REFERENCES

1. PRELIMINARY STUDY ON WILCOXON LEARNING MACHINES BY HSEIH,LIN, AND JENG IEEE TRANSACTIONS ON NEURAL NETWORKS,VOL.19,NO.2 FEB 2008

2. IDENTIFICATION AND CONTROL OF DYNAMICAL SYSTEMS USING NEURAL NETWORKS BY KUMPATI S. NARENDRA FELLOW, IEEE. AND KANNAN PARTHASARATHY IEEE TRANSACTLONS ON NEURAL NETWORKS. VOL. I . NO. I . MARCH 1990

3. FILTERED-X RADIAL BASIS FUNCTION NEURAL NETWORKS FOR ACTIVE NOISE CONTROL BAMBANG RIYANTO, LAZUARDI ANGGONO & KENKO UCHIDA PROC. ITB ENG. SCIENCE VOL. 36 B, NO. 1, 2004, 21-42

4. A JOINT STOCHASTIC GRADIENT ALGORITHM AND ITS APPLICATION TO SYSTEM IDENTIFICATION WITH RBF NETWORKS BADONG CHEN, JINCHUN HU, HONGBO LI, AND ZENGQI SUN PROCEEDINGS OF THE 6TH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, JUNE 21 - 23, 2006, DALIAN, CHINA

5. "RADIAL BASIS FUNCTION NETWORKS "CHAPTER 20,P-855-874 ADAPTIVE FILTER THEORY BY SIMON HAYKIN PHI PUBLICATIONS

6. NEURAL NETWORKS: A COMPREHENSIVE FOUNDATION 2ND EDITION BY SIMON HAYKIN, PEARSON EDUCATION

7. NEURAL NETWORKS: A CLASSROOM APPROACH BY SATISH KUMAR, TATA MCGRAW HILL

8. A NEURAL NETWORK ENVIRONMENT FOR ADAPTIVE INVERSE CONTROL HELDER J. COCHOFEL, B.SC. DAN WOOTEN, M.SC. JOSE PRINCIPE, PH.D.

9. AN INTRODUCTION TO THE USEOF NEURAL NETWORKS IN CONTROL SYSTEMS MARTIN T. HAGAN, HOWARD B. DEMUTH AND ORLANDO DE JESÚS